RAIRO-Theor. Inf. Appl. **42** (2007) 237–252
DOI: 10.1051/ita:2007032

Available online at:
www.rairo-ita.org

PARALLEL APPROXIMATION TO HIGH MULTIPLICITY SCHEDULING PROBLEMS *VIA* SMOOTH MULTI-VALUED QUADRATIC PROGRAMMING *, **

Maria Serna[1] and Fatos Xhafa

Abstract. We consider the parallel approximability of two problems arising from high multiplicity scheduling, namely the *unweighted model with variable processing requirements* and the *weighted model with identical processing requirements*. These two problems are known to be modelled by a class of quadratic programs that are efficiently solvable in polynomial time. On the parallel setting, both problems are P-complete and hence cannot be efficiently solved in parallel unless P = NC. To deal with the parallel approximablity of these problems, we show first a parallel *additive* approximation procedure to a subclass of multi-valued quadratic programming, called smooth multi-valued QP, which is defined by imposing certain restrictions on the coefficients of the instance. We use this procedure to obtain parallel approximation to *dense* instances of the two problems by observing that dense instances of these problems are instances of smooth multi-valued QP. The *dense* instances of the problems considered here are defined similarly as for other combinatorial problems in the literature. For such instances we can find in parallel a near optimal schedule. The definition of smooth multi-valued QP as well as the procedure for approximating it in parallel are of interest independently of the application to the scheduling problems considered in this paper.

Mathematics Subject Classification. 68W10, 68W25, 90B35, 90C20.

Keywords and phrases. Parallel approximation, quadratic programming, multiplicity scheduling problem.

* *Preliminary version presented at Vector and Parallel Processing Meeting (VECPAR'00), Porto, Portugal, June 2000.*

** *Research partially supported by ASCE Project TIN2005-09198-C02-02 and Project FP6-2004-IST-FETPI (AEOLUS).*

[1] Department of LSI, Universitat Politècnica de Catalunya, Campus Nord, Ed. Omega, C/Jordi Girona Salgado, 1-3, 08034-Barcelona, Spain; fatos@lsi.upc.edu

Article published by EDP Sciences

1. Introduction

In high multiplicity scheduling problems, jobs are partitioned into groups (or *types*) and in each group all the jobs are identical. The number of jobs of a certain type is called the multiplicity of that type. The goal is to find a schedule that minimizes a specified parameter such as completion time, lateness, tardiness, etc. (see *e.g.*, [8] for definitions and known results on different subclasses of the problem.) multiplicity scheduling problems, in their general form, are NP-hard. In fact, the problem remains NP-hard even for the simple case where there is only one job of each type and there are two identical machines released at time zero, having processing capacity n (see, *e.g.* [5]). However, several subclasses of interest are obtained by restricting to the model where all job types have the same processing requirements. Among others, there are the *unweighted model with variable processing requirements* and the *weighted model with identical processing requirements*. In [6] are given polynomial time algorithms for these problems by modelling them as convex separable QP. (Even strongly polynomial time[1] algorithms are known [8] for the case of a single machine.)

In the parallel setting, both problems are P-complete since the *general list scheduling problem*, which is known to be P-complete (see [7]), can be reduced to both problems. The algorithm of [6] cannot be efficiently parallelized[2], unless P=NC, since convex-separable quadratic programs are shown to be even non-approximable in parallel [17].

To the best of our knowledge the parallel approximability of the two problems has not been previously considered. We address the parallel approximability for *"dense"* instances[3] of the two problems. The definition of a dense instance for these problems is done similarly as for other combinatorial optimization problems in the literature [3,4,9–11]. Such instances have minimum completion time $\Omega(N^2)$, where N denotes the total number of jobs and satisfy some restrictions on the weights, the released times, as well as on the processing times of the jobs. When restricting to dense instances, we are able to efficiently find in parallel a near optimal schedule, that is, a schedule of jobs into machines whose completion time is at most $(1+\varepsilon)$ times the minimum schedule.

We define a subclass of multi-valued quadratic programming, called smooth multi-valued QP, by imposing certain restrictions on the coefficients of the instance. Next we show a parallel *additive* approximation procedure for smooth multi-valued QP, and then we use this procedure to obtain parallel approximation to *dense* instances of the two problems mentioned above by observing that dense instances of these problems are instances of smooth multi-valued QP. Our definition of smooth multi-valued QP as well as the proposed procedure for

[1] A strongly polynomial time algorithm runs in polynomial time whenever the arithmetic operations can be done in polynomial time.

[2] The parallel complexity class NC consists of those problems that can be solved very fast in parallel.

[3] It is not known whether these two problems remain P-complete when restricting to dense instances.

approximating it in parallel are of interest independently of the application to the scheduling problems considered in this paper.

Smooth quadratic programming (Smooth QP) in 0/1 variables was first defined by Arora *et al.* [4] by imposing restrictions on the magnitudes of the coefficients appearing in the instances of the problem. Smooth QP was shown to have an additive approximation procedures in polynomial time, *i.e.*, a procedure that finds in polynomial time approximate solutions whose objective function value are within an additive error from the optimum value [3,4]. Interestingly, there is a close relation between *smooth* instances of QP and *dense* instances of several combinatorial optimization problems. Indeed, it was shown in [3,4] that many combinatorial optimization problems can be casted by quadratic programs and the QPs corresponding to their dense instances are smooth instances. From this connection were obtained polynomial times approximation Schemes for dense instances of several NP-hard problems including Max CUT, Max kSAT, linear arrangements problems, etc.

The approximability of Smooth QP has been also considered in the parallel setting [18] where it was proven that the scheme of [4] can be also done in parallel. It should be mentioned, however, that the Smooth QP considered in [3,4] are in 0/1 variables. Clearly, a larger subclass of QP is that of multi-valued integer variables, we call this class smooth multi-valued QP (Smooth MQP). We extend the result on Smooth QP in 0/1 variables to the smooth multi-valued QP by showing that there is a parallel additive approximation procedure to the instances of the problem. The extension to the multi-valued case is done by reducing in NC the instance of QP to an instance of LP in packing/covering form whose near-optimal solution can be found in NC through the algorithm of Luby and Nisan [12] and then the fractional solution is rounded to an integer one.

The paper is organized as follows. In Section 2 we formally define the problems used through the paper and briefly recall some known techniques that we make use of. The approximation procedure for smooth multi-valued QP is given in Section 3 and, in Section 4 we apply the approximation procedure to the two problems arising from high multiplicity scheduling introduced above. We conclude with some open questions.

2. Preliminaries and definitions

We recall first some basic notions from the parallel complexity theory. The main complexity classes in parallel computation are P, NC and the class of P-complete problems. Problems in P are problems that can be solved easily on a single processor with running time polynomial in instance size. NC is the class of the problems that admit an efficient parallel algorithm, that is, an algorithm running in poly logarithmic time and using a polynomial number of processors. It is generally believed that P-complete problems are outside of NC class, they are called intractable or hard to solve in parallel.

We use the notion of approximation to optimization problems in the parallel setting. An *NC approximation algorithm* of performance ratio r is an algorithm that runs in time polylog in the input size, uses a polynomial number (in the input size) of processors, and finds a feasible solution to the problem whose measure is guaranteed to be within the multiplicative factor r of the optimum. An *NC approximation scheme* (NCAS) is a family of NC algorithms $\{\mathcal{A}_\varepsilon\}_{\varepsilon \geq 0}$, such that $\mathcal{A}_\varepsilon$ has performance ratio $1+\varepsilon$, and whose running time depends arbitrarily on ε. The NCAS class consists of all problems for which there exist an NC approximation scheme.

We give now the definition of the smooth multi-valued QP that is central to this paper.

Definition 2.1 (smooth MQP)**.** *Let $A = (a_{ij})$ be an $n \times n$ matrix, $W = (w_{ij})$ an $m \times n$ matrix, b an n-vector and d an m-vector, over rational numbers and a is a rational number. Multi-valued quadratic programming (MQP) is*

$$\text{(Smooth MQP)} \quad \begin{cases} \min & \sum a_{ij}x_ix_j + \sum b_ix_i + a \\ \text{s.t.} & \\ & Wx \geq d \\ & 0 \leq x_i \leq c_i,\ 1 \leq i \leq n \\ & x_i \text{ integral},\ 1 \leq i \leq n. \end{cases}$$

An instance of MQP is called smooth if a_{ij} are $O(1)$, b_i are[4] *0 or $\Theta(n)$, a is 0 or $\Theta(n^2)$ and w_{ij} are $O(1)$. The entries of W and d are non-negative; c_i's are constants.*

The definition of Smooth MQP intents to capture subclasses of the MQP problem that represent advantages with respect to the approximability. Clearly, the imposed conditions restrict the problem, yet Smooth MQP will be strong enough to cast instances of interest for several combinatorial optimization problems, in particular, problems arising from high multiplicity scheduling problems.

In proving the approximability result we make use of the following known techniques. The first one is a standard technique for the estimation of the sum of n numbers by random sampling[5] (see, Lem. 2.1 in [2]).

Lemma 2.2 (sampling lemma)**.** *Let $\{a_i\}_{i=1}^n$ be a set of n numbers, where each a_i is a constant $O(1)$. Let $p = \sum_{i=1}^n a_i$ be their sum. If we pick uniformly at random a subset of $s = O(\log n/\varepsilon^2)$ of a_i's and compute their sum q, then with high probability, i.e., with probability at least $1 - O(1/n)$, we have that $p - \varepsilon n \leq qn/s \leq p + \varepsilon n$.*

The second technique that we use is the Randomized Rounding in NC by Alon and Srinivasan [1] to a subclass of linear programs. This can be seen as a parallel

[4] Some of the b_i's as well as a may not appear in the objective function, *i.e.*, they are equal to 0.

[5] There is also a version of this lemma by Arora et al. in which a_i are each with absolute value at most constant M. In [4] authors use yet another version of the lemma with $a_i \in \{0, 1\}$.

counterpart of the randomized rounding of Raghavan and Thompson [16] and Raghavan [15] that work in the sequential setting.

Raghavan's bound. Raghavan's result can be stated as follows: if v^* is the optimal fractional value of a packing integer program instance then an integral feasible solution to that instance can be found whose objective function value is $v' = v^*(1-\delta)$, for a small $\delta > 0$.

It should be noted that the scheme of Alon and Srinivasan can additionally deal with the more general case of packing integer programs having multi-valued integer variables.

Definition 2.3 (PIP)**.** *A packing integer program is to maximize $q^T x$ subject to $Mx \leq p$ where $M \in [0,1]^{m \times n}$, p is an m-vector, and q is an n-vector such that the entries of p and q are non-negative rational numbers, with the integrality constraint on variables $x_j \in \{0, 1, \ldots, d_j\}$; some of d_j could be also infinite.*

Essentially, the Alon and Srinivasan's technique starts from a fractional solution of the linear program and shows how to do the rounding efficiently in NC. The integer feasible solution is also found *deterministically* in NC. Their result is summarized in the next two theorems.

Theorem 2.4. *[1] Given an instance of PIP, if the right hand sides p_i are constants bounded by $O(\log(m+n))$ then it can be approximated in NC to within a (1+o(1)) factor of the Raghavan's sequential bound [15].*

There are applications, however, in which simply an integer feasible solution of the PIP might be required. In such a case the following version of the above theorem can be applied. Note that in this case we lose the "reasonable" performance guarantee of the feasible integral solution.

Theorem 2.5. *[1] For any constant $c > 1$, PIPs can be approximated in NC to within $1/c$ factor of the Raghavan's sequential bound [15].*

We note that PIPs can be seen as special cases of a more general class of linear programs known in literature as *positive linear programming* introduced by Luby and Nisan.

Definition 2.6 (positive linear programming (PLP) [12])**.** *A maximization linear program is said to be an instance of* positive linear programming *if it is written as* $\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ *where all the entries of A, $\mathbf{b}$ and $\mathbf{c}$ are non-negative.*

Luby and Nisan developed a very efficient algorithm for approximating positive linear programming problems.

Theorem 2.7. *[12] There exists a parallel algorithm that given in input a maximization instance P of PLP and a rational $\varepsilon > 0$ returns a feasible solution for P whose cost is at least $(1-\varepsilon)$ times the optimum. Furthermore, the algorithm runs in time polynomial in $1/\varepsilon$ and $\log N$ using $O(N)$ processors, where N is the number of non-zero entries in P.*

3. Approximating smooth multi-valued quadratic programming

In this section we show an NC additive approximation procedure that, given in input an instance of Smooth MQP, finds a feasible solution x to that instance whose objective function value is within an additive error from the optimum value. In other terms, by letting $g(\cdot)$ the objective function and x^* the optimal solution, it holds that $g(x) \leq g(x^*) + \varepsilon n^2$, where n is the number of variables and ε a positive constant. As a corollary, when the optimal value $g(x^*)$ of the MQP instance is $\Omega(n^2)$, we obtain an $(1+\varepsilon)$-approximation for any (constant) value of $\varepsilon > 0$, thus the problem has an NC approximation scheme. In particular, we obtain NCAS for positive instances of Smooth MQP since, as we will show, for such instances $g(x^*) = \Omega(n^2)$.

We prove the following theorem.

Theorem 3.1. *Given an instance of Smooth MQP such that $Wx \geq d$, $0 \leq x_i \leq c_i$, $1 \leq i \leq n$, is feasible and a fixed $\varepsilon > 0$, we can find*[6] *in NC an integral vector y with $0 \leq y_i \leq c_i$, such that y satisfies $Wy \geq d$, and*

$$g(y_1, y_2, \ldots, y_n) \leq g(x^*) + \varepsilon n^2, \tag{1}$$

where $g(\cdot)$ is the objective function and $g(x^*)$ is its optimal value.

For ease of exposition and notation we present the result for the positive instances of Smooth MQP, *i.e.* we suppose without loss of generality that the instance has all the coefficients non-negative and then we show how to deal with the general case in Section 3.4. We refer to such instances as *positive instances*.

3.1. Approximating positive smooth MQP

We state now Theorem 3.1 for the Positive Smooth MQP.

Theorem 3.2. *Given a positive instance of Smooth MQP such that $Wx \geq d$, $0 \leq x_i \leq c_i$, $1 \leq i \leq n$, is feasible and a fixed $\varepsilon > 0$, we can find in NC an integral vector y with $0 \leq y_i \leq c_i$, such that y satisfies $Wy \geq d$, and*

$$g(y_1, y_2, \ldots, y_n) \leq g(x^*) + \varepsilon n^2, \tag{2}$$

where $g(\cdot)$ is the objective function and $g(x^*)$ is its optimal value.

The proof of the theorem follows a similar theorem in [18] for Smooth MQP with 0/1 variables. One could try to reduce the multi-valued case to the 0/1 variable case by using a binary representation of variables x_i and then derive the result from the theorem in [18].

[6] The precise bounds on parallel running time and number of processors are determined by the Alon and Srinivasan's theorem, which are stated in NC-like terms. These bounds are essentially the ones given by Luby and Nissan algorithm for positive linear programs.

There are, however, two drawbacks with such approach: first, the number of variables of the program would grow by the total sum of all c_i (each x_i will be represented as a sum of c_i 0/1 variables) impacting thus in the instance size; secondly, we have to yet prove the equivalence of the two programs – the Smooth MQP with multi-value variables and Smooth MQP with 0/1 variables, which is not straightforward[7] at all. Hence, we prove the theorem by mimicking the steps for the 0/1 case.

To prove the theorem, we write the program (Smooth MQP) equivalently as[8]:

$$\begin{cases} \min & c; \\ \\ \text{s.t.} & x^T Ax + bx \le c, \\ & Wx \ge d, \\ & 0 \le x_i \le c_i,\ 1 \le i \le n, \\ & x_i \text{ integral},\ 1 \le i \le n. \end{cases}$$

Notice that, by using a binary search, for Theorem 3.2 it suffices to prove the following theorem.

Theorem 3.3. *Suppose there is an integral solution to the following positive smooth multi-valued quadratic system*

$$\text{(Positive Smooth MQS)} \quad \begin{cases} x^T Ax + bx \le c \\ \\ Wx \ge d \\ \\ 0 \le x_i \le c_i,\ 1 \le i \le n. \end{cases}$$

Then, for any fixed $\varepsilon > 0$, we can find in NC an integer vector y with $0 \le y_i \le c_i, 1 \le i \le n$, that satisfies $Wy \ge d$ and such that

$$y^T Ay + by \le c + \varepsilon n^2. \tag{3}$$

The main steps of the proof are as follows:

1. positive linear programming instance: reduce in parallel the instance of (Positive Smooth MQS) to an instance of positive linear programming.
   - 1a.: write the (Positive Smooth MQS) instance as an appropriate linear program by relaxing the integrality conditions on x_i's; we denote it by (LP1);
   - 1b.: transform the instance of linear program (LP1) into a positive linear program through instances of linear programs (LP2) and (LP3) (see below).

[7] Usually results for multi-valued integer programs are directly proved rather than derived from known results for the 0/1 case; the Alon and Srinivasan's theorem on PIPs could be considered as such an example.

[8] The constant a in the objective function is unimportant.

2. Fractional solution: invoke the Luby and Nisan's procedure to find in parallel an approximate (sub-optimal) fractional solution to the instance of the linear program (LP3).
3. Rounding of the fractional solution to an integer solution: round in parallel the fractional solution of Step 2 to an integer solution through the technique of Alon and Srinivasan.

We deal with these steps in the following two subsections.

3.2. Reducing positive smooth MQS to positive LP

Let $\overline{x}$ be a feasible solution to (Positive Smooth MQS), as supposed, and let us write $\overline{r} = \overline{x}A + b$, where the components of $\overline{r}$ are computed (estimated) using sampling lemma (see Lem. 2.2 and [2] for details). Notice that $\overline{r}_i = \Theta(n)$ due to the conditions on the coefficients (a_{ij} are positive $O(1)$ values and b_i are 0 or $\theta(n)$; see Def. 2.1) and the definition of sampling[9] values r_i. Since $x^T Ax + bx = (x^T A + b)x$, we can express the quadratic program (Positive Smooth MQS) as a linear program as follows:

$$\text{(LP1)} \quad \begin{cases} x^T A + b \leq \overline{r}, \\ \overline{r}x \leq c, \\ Wx \geq d, \\ 0 \leq x_i \leq c_i, \qquad 1 \leq i \leq n, \end{cases}$$

for which $\overline{x}$ is also a feasible solution. The following observation is immediate.

Proposition 3.4. *If x is a feasible solution to (LP1) then x is also feasible to (Positive Smooth MQS).*

Clearly, any coefficient in (LP1) is non-negative. However, (LP1) is not a positive linear program because in it we have both types of restrictions $\geq$ and $\leq$. To overcome this, we modify (LP1) appropriately by introducing variables $z_i = c_i - x_i$. Thus, (LP1) is written as:

$$\text{(LP2)} \quad \begin{cases} x^T A + b \leq \overline{r}, & \\ \overline{r}x \leq c, & \\ \sum_{j=1}^{n} w_{kj} z_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k, & 1 \leq k \leq m, \\ x_i + z_i = c_i, & 1 \leq i \leq n, \\ 0 \leq x_i, z_i, & 1 \leq i \leq n, \end{cases}$$

We can suppose, without loss of generality, that $\forall k$, $1 \leq k \leq m$, $\sum_{j=1}^{n} w_{kj} c_j - d_k \geq 0$ because otherwise the system $\{Wx \geq d,\ x_i \leq c_i,\ 1 \leq i \leq n\}$ would not be feasible. So, the above program (LP2) is still positive. The relation between (LP1) and (LP2) is given as follows.

Proposition 3.5. *(a) If x is a feasible solution to (LP1) then (x,z), where $z_i = c_i - x_i$, is also feasible to (LP2); (b) if (x,z) is a feasible solution to (LP2) then x is a feasible solution to (LP1).*

[9] The values r_i used in sampling are defined as $r_i = b_i + (n/k) \cdot \sum_{j \in S} a_{ij} s_j$ where S is a set of $k = O((\log n)/\varepsilon^2)$ indices.

Finally, to transform the conditions $x_i + z_i = c_i$ into $x_i + z_i \leq c_i$, we add to (LP2) an adequate objective function resulting in the following program:

$$\text{(LP3)} \quad \begin{cases} \max & \sum_{i=1}^{n}(x_i + z_i); & \\ \text{s.t.} & x^T A + b \leq \overline{r}, & \\ & \overline{r}x \leq c, & \\ & \sum_{j=1}^{n} w_{kj} z_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k, & 1 \leq k \leq m, \\ & x_i + z_i \leq c_i, & 1 \leq i \leq n, \\ & 0 \leq x_i, z_i, & 1 \leq i \leq n. \end{cases}$$

Notice that (LP3) is a positive linear program, as required by the Luby and Nisan's algorithm. The programs (LP2) and (LP3) are related as follows:

Proposition 3.6. *(a) If (x, z) is a feasible solution to (LP2) then it is also feasible to (LP3); (b) if (x, z) is an $(1 - \varepsilon)$-***optimal** *solution to (LP3) then we can construct (x', z') in NC such that it is* **feasible** *to $\{\sum_{j=1}^{n} w_{kj} z_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k$, $1 \leq k \leq m,\ x_i + z_i \leq c_i,\ 1 \leq i \leq n\}$ and*

$$\begin{cases} x'^T A + b \leq \overline{r} + K_1 \varepsilon \mathbf{1} \\ \overline{r}x' \leq c + K_2 \varepsilon n \end{cases}$$

where K_1 and K_2 are two constants computable in NC from the instance and $\mathbf{1}$ *is the vector all whose entries are equal to* 1.

Proof. b) A near-optimal solution (x, z) to (LP3) can be found *via* the Luby and Nisan's algorithm. Such solution will have cost at least $(1-\varepsilon)\sum_{i=1}^{n} c_i$ (supposing that (LP2) is feasible). Now, we define $z' = z$ and x' by taking $x'_i = c_i - z_i$. Since, due to near optimality of the solution (x, z),

$$\sum_{i=1}^{n}(x_i + z_i) \geq (1 - \varepsilon)\sum_{i=1}^{n} c_i$$

we have that $x_i + z_i \approx (1 - \varepsilon)c_i$ therefore, intuitively, the values of x'_i will not increase too much. More precisely, let, for any i, $\varepsilon_i = x'_i - x_i$ and also let $e = (\varepsilon_1, \ldots, \varepsilon_n)$. Notice that ε_i are non-negative values. By the definition of ε_i we have:

$$\begin{aligned} \sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n}(x'_i - x_i) &= \sum_{i=1}^{n} c_i - \sum_{i}^{n}(x_i + z_i) \\ &\leq \sum_{i=1}^{n} c_i - (1 - \varepsilon)\sum_{i=1}^{n} c_i = \varepsilon \sum_{i=1}^{n} c_i. \end{aligned}$$

Clearly, the system

$$\left\{ \sum_{j=1}^{n} w_{kj} z_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k,\ 1 \leq k \leq m,\ x_i \leq c_i,\ 1 \leq i \leq n \right\}$$

is satisfied by (x', z'). We check now whether it satisfies the restrictions (4). Due to the magnitudes of $a_{ij} = O(1)$ and $\overline{r}_i = \Theta(n)$, we have that

$$x'^T A + b = (x^T + e)A + b \leq (x^T A + b) + eA \leq \overline{r} + K_1 \varepsilon \mathbf{1} \tag{4}$$

$$\overline{r} x' = \overline{r}(x + e) \leq c + K_2 \varepsilon n \tag{5}$$

where K_1 and K_2 are two constants defined as follows: $K_1 = (\sum_{i=1}^{n} c_i) \cdot \max_{i,j} a_{ij}$ and $K_2 = (\sum_{i=1}^{n} c_i) \cdot \max_i \alpha_i$ where α_i are the upper bounds on $\overline{r}_i$, $\overline{r}_i \leq \alpha_i \cdot n$. □

3.3. Rounding of the fractional solution to an integer solution

Having the fractional solution (x', z'), as given in Proposition 3.6, we apply the randomized rounding procedure of Alon and Srinivasan to the system

$$\left\{ \sum_{j=1}^{n} w_{kj} z_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k,\ 1 \leq k \leq m \right\}.$$

Rounding the feasible fractional solution z' gives an integral solution u, that satisfies

$$\sum_{j=1}^{n} w_{kj} u_j \leq \sum_{j=1}^{n} w_{kj} c_j - d_k,\ 1 \leq k \leq m.$$

Letting y such that $y_i = c_i - u_i$, we have

$$\begin{aligned} &Wy \geq d, \\ &y_i \leq c_i, \qquad 1 \leq i \leq n. \end{aligned} \tag{6}$$

Furthermore, for y we have:

$$\begin{aligned} &y^T A_i + b_i \leq (\overline{r}_i + K_1 \varepsilon \mathbf{1}) + O(\sqrt{n \log n}), \\ &\overline{r} y \leq (c + K_2 \varepsilon n) + O(n) \cdot O(\sqrt{n \log n}). \end{aligned} \tag{7}$$

Consequently,

$$
\begin{aligned}
y^T Ay + by &= (y^T A + b)y \leq \left((\overline{r} + K_1\varepsilon) + O(\sqrt{n\log n})\right) y \\
&\leq \overline{r}y + K_1\varepsilon \mathbf{1}\cdot y + O(\sqrt{n\log n})\mathbf{1}\cdot y \\
&\leq (c + K_2\varepsilon n) + O(n)\cdot O(\sqrt{n\log n}) + K_1\varepsilon\mathbf{1}\cdot y + O(\sqrt{n\log n})\mathbf{1}\cdot y \qquad (8) \\
&\leq c + (\varepsilon + o(1))n^2 + O(n^{3/2}\sqrt{\log n}) + (\varepsilon + o(1))n^2 + O(n^{3/2}\sqrt{\log n}) \\
&\leq c + (2\varepsilon + o(1))n^2,
\end{aligned}
$$

where the last inequality is derived by using the definition of K_1 and K_2 (see Prop. 3.6) and the fact that y has integral non-negative values; we have then upper bounded each term (except c) by n^2. Thus, by taking $\varepsilon' = (2\varepsilon + o(1))$ we obtain the solution (x', z') that yields to the solution y satisfying (6) and $y^T Ay + by \leq c + \varepsilon' n^2$.

But, we can write (LP1) only if we knew the values $\overline{r}_i$, *i.e.*, the vector $\overline{r}$. Instead, it is shown in [4] that using estimates r_i for them such that $|\overline{r}_i - r_i| < \varepsilon n$ then (7) and (8) still hold. These estimates are found similarly as in [3,4], through the Sampling Lemma, and we omit the details.

Derandomization. Notice that the proof of Theorem 3.3 involves sampling lemma (Step 1) and the Rounding of the fractional solution (Step 3). The de-randomization of sampling lemma can be found in [18]. As regards the rounding of the fractional solution, Alon and Srinivasan [1] theorem already provides the rounding procedure deterministically in NC.

Corollary 3.7. *If the instance of Smooth MQP has optimal value $g(x^*) = \Omega(n^2)$, i.e., $g(x^*) \geq \delta n^2$, for some $\delta > 0$, then Theorem 3.1 implies an $(1+\varepsilon)$-approximation to such instances, i.e. an NC approximation scheme.*

Indeed, in this case the solution x satisfies $g(x) \leq (1 + \varepsilon/\delta)g(x^*)$.

3.4. The general case: smooth MQP

A close observation of the proof of Theorem 3.2 shows that the assumption on the coefficients a_{ij}, b_i and a of the MQP objective function to be non-negative can be removed[10]. Indeed, this assumption is used only in the reduction step to obtain a positive linear program (LP3) in order to enable the application of Luby and Nisan's algorithm. More precisely, the positiveness assumption assures that

[10] To be precise, the notation on the magnitudes of the coefficients given in the definition of Smooth MQP becomes now $a_{ij} = O(1)$, $b_i = O(n)$ and $c = O(n^2)$; also, $r_i^{\#} = O(n)$ (the notation used in [4]).

the linear restrictions

$$\begin{aligned} x^T A + b &\leq r^{\#} \\ r^{\#} x &\leq c \end{aligned} \tag{9}$$

have positive coefficients. This last fact can be assured without the positiveness assumption as follows:

a) if some $a_{ij} < 0$ then substitute the term $a_{ij}x_i$ by $-a_{ij}(z_i - c_i)$;
b) if some $r_i^{\#} < 0$ then substitute the term $r_i^{\#}x_i$ by $-r_i^{\#}(z_i - c_i)$.

Notice that by applying a) the left-hand sides of restrictions (9) become non-negative and therefore the right-hand sides so do (otherwise the program would be infeasible). Doing these (possible) variable changes we obtain the positive linear program (LP3). The variable change effects the right-hand sides of $\{x^T A + b \leq r^{\#}, r^{\#}x \leq c\}$ but yet it doesn't effect Proposition 3.6 due to the definition of (x', z') and the magnitudes of the coefficients. On the other hand, removing the positiveness condition on a_{ij}, b_i and a doesn't effect the rounding scheme of Alon and Srinivasan since it applies to the program $Wz \leq W \cdot c - d$. From this observation we can restate Theorem 3.2 without the positiveness condition on the coefficients of the objective function of the MQP instance thus yielding Theorem 3.1.

4. Applications

In this section we show that the additive approximation procedure for Smooth MQP can be applied to a couple of problems arising from high multiplicity scheduling problems, namely, the *weighted model with identical processing requirements* and the *unweighted model with variable processing requirements*. We first give the description of the general model of high multiplicity scheduling.

General model

The multiplicity scheduling problem, in its general form, is stated as follows [6]. There are n types of jobs, $J_1, J_2, \ldots, J_n$ and there are n_j identical jobs of type J_j. There are m parallel machines $M_1, M_2, \ldots, M_m$ to process the jobs. Machine M_i is released at time r_i and it can process at most c_i jobs. Each job should be processed in its entirety in one of the m machines. All the jobs are available for processing at time 0. The processing requirement of job of type j on machine i is p_{ij} time units. There is a non-negative weight w_{ij} associated to job j when processed by machine i. The objective is to schedule the jobs on m machines so that the total weighted completion time is minimized.

Weighted model with identical processing requirements

In this case we have identical processing requirements (the p_{ij} are written as $p_{ij} = 1/s_i$) and general weights w_{ij}. For any machine i, $i = 1, \ldots, m$, let σ_i be a permutation of $\{1, \ldots, n\}$ which arranges n types of jobs in a non-increasing order of their weights. Let also the variable x_{ij} denote the number of jobs of type $\sigma_i(j)$ processed in machine i. The cost of assigning a job of type $\sigma_i(j)$ to the kth place, $k = 1, \ldots, c_i$, is $(r_i + k(1/s_i))w_{i,\sigma_i(j)}$. The integer programming formulation [6] of the problem (WIP) follows.

$$
\text{(WIP)} \begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} r_i w_{i\sigma_i(j)} x_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{s_i} w_{i\sigma_i(j)} \sum_{k=1}^{x_{ij}} \left(\sum_{l=1}^{j-1} x_{il} + k \right) \\ \text{s.t.} \\ \qquad \sum_{j=1}^{n} x_{ij} = c_i, \ (i = 1, \ldots, m) \\ \qquad \sum_{\{(i,k), i=1,\ldots m, \sigma_i(k)=j\}} x_{ik} = n_j, \ (j = 1, \ldots, n) \\ \qquad 0 \le x_{ij} \le c_i, \text{integral}, \ (i = 1, \ldots, m; \ j = 1, \ldots, n). \end{cases}
$$

Notice that the restriction $\sum_{j=1}^{n} x_{ij} = c_i, (i = 1, \ldots, m)$ is written with equality since we can always introduce an additional type of job $(n + 1)$ with weight $w_{i,n+1} = 0$. In [6] this program was written as a convex separable quadratic program and then was solved through known polynomial time algorithms (*e.g.* [13]).

Unweighted model with variable processing requirements

This problem is obtained from the general model by letting $w_{ij} = v_i$ for all types of job j and machine i. The processing time of each one of the n_j jobs of type j on the ith machine is p_{ij}. Let us suppose that, for any machine i, $i = 1, \ldots, m$, the processing times are sorted in non-increasing order given by the permutation σ_i. Let also the variable x_{ij} denote the number of jobs of type $\sigma_i(j)$ processed in machine i. With these notations, we have the following integer programming formulation [6] for the problem (UIP).

$$
\text{(UIP)} \begin{cases} \min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} r_i v_i x_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} v_i p_{i\sigma_i(j)} \sum_{k=1}^{x_{ij}} \left(\sum_{l=1}^{j-1} x_{il} + k \right) \\ \text{s.t.} \\ \qquad \sum_{j=1}^{n} x_{ij} = c_i, \ (i = 1, \ldots, m) \\ \qquad \sum_{\{(i,k), 1 \le i \le m, \sigma_i(j)=k\}} x_{ij} = n_j, \ (j = 1, \ldots, n) \\ \qquad 0 \le x_{ij}, \text{integral}, \ (i = 1, \ldots, m; \ j = 1, \ldots, n). \end{cases}
$$

4.1. Approximation results

In order to apply the additive approximation procedure, we limit ourselves to *dense* instances of both problems as specified below. We say that an instance of

the high multiplicity scheduling problem (weighted model with identical processing requirements and unweighted model with variable processing requirements) is *dense* if it satisfies the following two restrictions: (a) the weights w_{ij}, the released times r_i and the processing times p_{ij} are bounded by constants; (b) the completion time of the minimum schedule is $\Omega(n^2)$.

Further, we consider a relaxation of the integer program (WIP) by relaxing the equality restrictions into covering type restrictions by adding *dummy* variables as follows.

$$
(\mathsf{WQP})\ \begin{cases}
\min & \sum_{i=1}^{m}\sum_{j=1}^{n} r_i w_{i\sigma_i(j)} x_{ij} \\
& + \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{1}{s_i} w_{i\sigma_i(j)}\big(x_{ij}(x_{i1}+\cdots+x_{i,j-1}+\frac{1}{2}x_{ij})\big) \\
& + \sum_{i=1}^{m} L_i^{(1)} \cdot y_i + \sum_{j=1}^{n} L_j^{(2)} \cdot z_j \\
\text{s.t.} & \\
& \sum_{j=1}^{n} x_{ij} + y_i \geq c_i,\ (i=1,\ldots,m) \\
& \sum_{\{(i,k),i=1,\ldots m,\sigma_i(k)=j\}} x_{ik} + z_j \geq n_j,\ (j=1,\ldots,n) \\
& 0 \leq x_{ij}, y_i \leq c_i,\ (i=1,\ldots,m;\ j=1,\ldots,n) \\
& 0 \leq z_j \leq n_j,\ j=1,\ldots,n,
\end{cases}
$$

where $L_i^{(1)}$ $(i=1,\ldots,m)$ and $L_j^{(2)}$ $(j=1,\ldots,n)$ are positive constants defined next. The programs (WIP) and (WQP) are equivalent with regard to the optimal and near-optimal solutions. To this end, the term added to the objective function due to the relaxation is multiplied by appropriately chosen positive constants $L_i^{(1)}$ and $L_j^{(2)}$. This kind of equivalence between a program and its relaxed version with regard to optimality and near-optimality of solutions was first given in [20,21] and further used in [22] and [19]. It consists essentially in defining the multiplying coefficients $L_i^{(1)}$ and $L_j^{(2)}$ as a sum of the "weights" of constraints where the introduced variables y_i and z_j appear, resp. More precisely, we define, for any $i=1,\ldots,m$, $L_i^{(1)} = \sum_j w_{i\sigma_i(j)}$ and for any $j=1\ldots n$, $L_j^{(2)} = \sum_{\{(i,k),i=1,\ldots m,\sigma_i(k)=j\}} w_{i\sigma_i(k)}$. It should be noticed that this choice of the multiplying coefficients is necessary for the near-optimality equivalence between the two programs; for the optimality equivalence large positive constants would suffice. The reader is refereed to [19,20] for further details.

The following relation between dense instances of our scheduling problem and the smooth instances of multi-valued QP is straightforward.

Proposition 4.1. *If the instance of scheduling is dense then the corresponding (WQP) instance is a smooth multi-valued QP instance.*

Hence by applying Theorem 3.1 and Corollary 3.7 to such instances we obtain the following theorem.

Theorem 4.2. *Given a dense instance of weighted models with identical processing requirements of N jobs ($N = \sum_{j=1}^{n} n_j$) and m machines there is an NC algorithm that finds a schedule whose completion time is at most $(1+\varepsilon)$ times the minimum schedule.*

We apply our approximation scheme to the dense instances of unweighted model with variable processing requirements in the same way as in the case of the weighted model with identical processing requirements and obtain the following theorem.

Theorem 4.3. *Given a dense instance of unweighted model with variable processing requirements of N jobs ($N = \sum_{j=1}^{n} n_j$) jobs and m machines there is an NC algorithm that finds a schedule whose completion time is at most $(1+\varepsilon)$ times the minimum schedule.*

Open questions

Our approximation procedure applies to a restricted class of instances of the scheduling problems. It would be interesting to extend the result to other instances of the problem without the denseness condition. Also, applying our procedure to other combinatorial optimization problems modelled by multi-valued QP would be of interest.

Acknowledgements. We thank Josep Díaz and Paul Spirakis for useful comments and discussion on issues related to parallel approximability of Linear Programming. Fatos Xhafa thanks Ray Greenlaw for an early discussion on the parallel complexity of quadratic programming problems.

References

[1] N. Alon, and A. Srinivasan, Improved parallel approximation of a class of integer programming problems. *Algorithmica* **17** (1997) 449–462.

[2] S. Arora, D. Karger, and M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems. Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing (*STOC '95*) **58** (1995) 284–293, ACM Press.

[3] S. Arora, A. Frieze, and H. Kaplan, A new rounding procedure for the assignment problem with applications to dense graph arrangement problems, in *Procedings of the FOCS'96* (1996) 21–30.

[4] S. Arora, D. Karger, and M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems. *J. Comput. Syst. Sci.* **58** (1999) 193–210.

[5] M.R. Garey, and D.S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co. (1979).

[6] F. Granot, J. Skorin-Kapov, and A. Tamir, Using quadratic programming to solve high multiplicity scheduling problems on parallel machines. *Algorithmica* **17** (1997) 100–110.

[7] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press (1995).
[8] D.S. Hochbaum, and R. Shamir, Strongly polynomial algorithms for the high multiplicity scheduling problem. *RAIRO Oper. Res.* **39** (1991) 648–653.
[9] M. Karpinski, J. Wirtgen, and A. Zelikovsky, An approximation algorithm for the bandwidth problem on dense graphs. Technical Report **TR97-017**, ECCC (1997).
[10] M. Karpinski, J. Wirtgen, and A. Zelikovsky, Polynomial times approximation schemes for some dense instances of NP-hard problems. Technical Report **TR97-024**, ECCC (1997).
[11] M. Karpinski, and A. Zelikovsky, Approximating dense cases of covering problems. Network design: connectivity and facilities location (Princeton, NJ, 1997). *Amer. Math. Soc.* (1998) 169–178.
[12] M. Luby, and N. Nisan, A parallel approximation algorithm for positive linear programming, in *Proceedings of 25th ACM Symposium on Theory of Computing* (1993) 448–457.
[13] M. Minoux, *Mathematical programming: theory and algorithms*, Wiley (1986).
[14] R. Motwani, and P. Raghavan, *Randomized Algorithms.* Cambridge University Press (1995).
[15] P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comput. Syst. Sci.* **37** (1988) 130–143.
[16] P. Raghavan, and C. Thompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* **7** (1987) 365–374.
[17] M. Serna, Approximating linear programming is logspace complete for P. *Inform. Process. Lett.* **37** (1991) 233–236.
[18] M. Serna, and F. Xhafa, The parallel approximability of a subclass of quadratic programming. *Theoret. Comput. Sci.* **259** (2001) 217–231.
[19] P.S. Efraimidis, and P.G. Spirakis, Fast, parallel and sequential approximations to "hard" combinatorial optimization problems. Technical Report **TR99/06/01**, CTI, Patras (June 1999).
[20] L. Trevisan, Luca Trevisan: Positive linear programming, parallel approximation and PCP's. *Lect. Notes Comput. Sci.* **1136** (1996) 62–75.
[21] L. Trevisan, Parallel approximation algorithms by positive linear programming. *Algorithmica* **21** (1998) 72–88.
[22] M. Serna, L. Trevisan, and F. Xhafa, The approximability of non-boolean satisfiability problems and restricted integer programming. *Theoret. Comput. Sci.* **332** (2005) 123–139.

Communicated by R. Baeza-Yates.
Received March 17, 2004. Accepted December 24, 2006.