Vector Assignment Schemes for Asymmetric Settings

Leah Epstein[*] Tamir Tassa [†]

All manuscript correspondence should be sent to:
Tamir Tassa,
Department of Mathematics and Computer Science,
The Open University,
P.O.Box 808,
Raanana 43107.
Phone: +972-9-7781254
Fax: +972-9-7780605
Email: tamir_tassa@yahoo.com

[*]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. `lea@idc.ac.il`. Research supported in part by the Israel Science Foundation, (grant no. 250/01)

[†]Division of Computer Science, The Open University, Raanana, Israel. `tamir_tassa@yahoo.com`.

Abstract

We consider off-line vector assignment problems. The goal is to assign input vectors to machines so that a given target function is minimized. The target function usually gives some measure of the quality of the distribution of input vectors among machines. In [8] we dealt with this problem where the cost function is symmetric; namely, all machines are identical. The dynamic programming techniques that were used in [8] do not extend to asymmetric settings. Here, we deal with an asymmetric setting where the cost functions per machine may be different for different machines. Using graph-based techniques, we design a polynomial time approximation scheme for a wide class of asymmetric target functions. Other than a significant extension of the class of cost functions to which our current scheme applies, the present PTAS is much simpler than the one in [8]. It should be noted that asymmetric cost functions appear very naturally in the so called line-up problem that motivated the study in [8] and the present study.

Keywords: scheduling, optimization, approximation schemes, layered graphs.

1 Introduction

A general framework for Vector Assignment Problems (VAPs) was presented in [8]. In a VAP, one is given a set of vectors,

$$I = \{\mathbf{x}^i \in (\mathcal{R}^+)^d : 1 \le i \le n\} \tag{1}$$

and aims at finding an assignment of those vectors to m machines,

$$A : \{1, \dots, n\} \to \{1, \dots, m\} \tag{2}$$

such that the value of some target function is minimized. Typical target functions take the form

$$F(A) = f(g(\boldsymbol{\ell}^1), \dots, g(\boldsymbol{\ell}^m)) \tag{3}$$

where

- $\boldsymbol{\ell}^k$, $1 \le k \le m$, are the corresponding load vectors on each of the machines,

$$\boldsymbol{\ell}^k = \sum_{A(i)=k} \mathbf{x}^i \quad , \quad 1 \le k \le m \ ;$$

- $g : (\mathcal{R}^+)^d \to \mathcal{R}^+$ is a function that evaluates the cost per machine; and
- $f : (\mathcal{R}^+)^m \to \mathcal{R}^+$ is a function that evaluates the final cost over all machines.

Such problems are usually NP-hard. Hence, polynomial time approximation schemes (PTAS) are sought. Such schemes produce a solution (i.e., an assignment (2)) whose cost is larger than that of an optimal solution by a factor of no more than $(1 + C\varepsilon)$, where $\varepsilon > 0$ is an arbitrary parameter and C is a constant independent of the input data (n, m and $\mathbf{x}^i$, $1 \le i \le n$) and ε; namely, if Φ^o is the optimal cost, the scheme produces a solution A that satisfies

$$F(A) \le (1 + C\varepsilon) \cdot \Phi^o \ . \tag{4}$$

The running time of such schemes is polynomial in n and m for every fixed ε and d.

This general framework encompasses most of the known scheduling, load balancing and assignment problems. In [8] we review some of the problems that are covered by this framework, *e.g.*, the makespan problem [10, 12], the ℓ_p minimization problem [1, 2], the extensible bin packing problem [5, 6] and the vector scheduling problem [4]. The reader is referred to that paper for a discussion of related problems and additional corresponding references. By considering the general setting (3), instead of the many private cases that were studied previously, and by carefully analyzing the necessary properties of the cost functions f and g, we were able to obtain a PTAS for a wide class of cost functions, using standard dynamic programming and linear programming methods. A novel idea introduced there enabled even a further generalization of our results: by replacing the small input vectors in I with other vectors of larger magnitude such that the impact on the value of the target function is small, we were able to design a PTAS even for the case where the inner cost function g is not monotone (a similar preprocessing step was used in [1]; however, the procedure there was much simpler since it was implemented for scalars). A typical family of non-monotone cost functions is that of the Sobolev norms; those norms take into account also an ℓ_p measure of the differences of the vector components. As described in [8], some applications call for such a non-monotone measure of the quality of the assignment.

The results of [8] strongly relied upon the symmetry assumption, i.e., the value of the target function is invariant under interchanges between the machines. We proceed to recall some of the ingredients of the core algorithm in [8] in order to clarify this dependency on the symmetry assumption. In the core algorithm, [8, Section 4.4], given an assignment of the input vectors to the machines and an approximation parameter ε, we approximate the load vector on each machine by a *Discretized Load Configuration*, or *DLC*, which is a pair of discretized vectors that approximate the sum of large input vectors and the sum of small input vectors on that machine. It is then shown that the number of possible DLCs, denoted by t, depends only on d, f, g and ε, and that by replacing the actual load vector with the sum of the two vectors in the DLC that represents it, the value of the cost function increases by a multiplicative factor of no more than $(1 + O(\varepsilon))$ (see Lemma 2 there). Letting Ω denote the set of all t possible DLCs, each assignment of DLCs from Ω to the m machines may be uniquely described by an *Assignment Configuration Vector*, or *ACV*, of the following form:

$$\mathbf{ACV} = (m_1, \ldots, m_t) \quad \text{where } 0 \le m_\tau \le m, \quad 1 \le \tau \le t \ \text{ and } \sum_{\tau=1}^{t} m_\tau = m$$

(m_τ denotes the number of machines that were assigned DLCs of type τ, $1 \le \tau \le t$). The number of ACVs is bounded by m^t. As t is a constant that depends solely on d, f, g and ε, the number of ACVs to consider is polynomial in m. Then, the algorithm proceeds to scan all possible ACVs in order to find the one that minimizes the cost function. This is the place where symmetry is essential. The ACVs only record how many machines were assigned a DLC of a specific type. This is a sufficient information in case the machines are identical. The number of possible assignments in such settings is bounded by the polynomial m^t. However, if the machines were not identical, the information encoded by an ACV would be insufficient. In such cases we would need to record the type of DLC per machine. The number of such configurations would be t^m, which is exponential in m.

The approach that we take here is entirely different and it does not present this obstacle. It enables us to extend the framework (3) to

$$F(A) = f(g^1(\boldsymbol{\ell}^1), \ldots, g^m(\boldsymbol{\ell}^m)) \ . \tag{5}$$

Namely, the per-machine cost evaluator, $g^k(\cdot)$, may be different for each machine (for example, it may depend on various machine parameters, such as capacity, speed or priority). In addition, the outer cost function f could be asymmetric.

We would like to point out that the application that was described in [8, Section 2] – the so-called *line up problem* – which motivated that study and the present one, may indeed give rise to asymmetric cost functions. In that problem, the input vectors represent different TV programs (e.g., CNN, PBS or the Sports Channel); more specifically, each vector represents the bit-rate of one TV program during some sampling period. The m machines, on the other hand, represent channels on which those programs should be transmitted to their destination. The TV programs need to be assigned to channels and, subsequently, all programs that were assigned to the same channel need to be multiplexed (i.e., merged together) and then "packed" into that channel. Since each TV program is characterized by a variable bit-rate while the channels are characterized by a fixed bit-rate, it is possible that at some time intervals the sum of bit-rates of the programs that were assigned to the same channel will be larger than the fixed bit-rate of that channel. When such an overflow occurs, it is necessary to apply compression to the input programs in order to make the packing possible. Such compression results in loss of data and, hence, quality degradation. The target function in this context is a measure of the output quality. The goal, then, is to find an assignment of programs to channels that, based on the sampled behavior of those programs, yields the best quality at the output. There are several target functions that may serve as an adequate measure to the output quality. Some of those functions depend on the capacities of the output channels. Hence, in settings that include channels of different capacities (this is the case in TV broadcast centers where different modulation methods are employed in different channels), the target function may be asymmetric.

We implement here a graph-based approach that is different from the dynamic and linear programming approach that was used in [8]. Graph based techniques were used in the literature in [11, 3, 7]. All these papers, however, deal with problems that are one-dimensional; namely, the items (jobs) to be scheduled in those problems are characterized by a scalar size. We generalize the method to deal with vectors. The improvements that are offered by the graph based approach, compared to the dynamic and linear programming approach in [8], are as follows:

1. The graph-based approach enables us to design a PTAS in the asymmetric setting.
2. It yields a much simpler algorithm than the one devised in [8]. While the latter was an iterative algorithm that implemented a binary search in order to approximate the optimal solution, the algorithm devised herein is not iterative and it arrives at the approximate solution in a single phase.
3. Our present results apply to a wider class of cost functions. More specifically, the outer cost function f need not be harmonic and the inner cost functions g^k need not be convex nor stable with respect to linear multiplications (see Section 2 for a definition and discussion of those properties).

Terminology and Notations. Throughout this paper we adopt the following conventions:

- The term *set* may refer also to *multisets* in some cases.
- Small case letters denote scalars; bold face small case letters denote vectors.
- A superscript of a vector denotes the index of the vector; a subscript of a vector indicates a component in that vector. E.g., $\boldsymbol{\ell}^k_j$ denotes the jth component of the vector $\boldsymbol{\ell}^k$.
- If $\gamma(k)$ is any expression that depends on k, then $f(\gamma(k))_{1\le k\le m}$ stands for $f(\gamma(1),\ldots,\gamma(m))$.
- If $\mathbf{x}$ and $\mathbf{y}$ are vectors and c is a scalar, then $\mathbf{x}\le\mathbf{y}$ and $\mathbf{x}\le c$ mean that the inequality holds component-wise.

The extended abstract [9] reports the main results of both [8] and this paper.

2 The Cost Functions

Herein we list the assumptions that we make on the outer cost function $f(\cdot)$ and the inner cost function $g(\cdot)$.

Definition 1

1. *A function $h:(\mathcal{R}^+)^n\to\mathcal{R}^+$ is monotone if*
$$h(\mathbf{x})\le h(\mathbf{y})\quad \forall \mathbf{x},\mathbf{y}\in(\mathcal{R}^+)^n \ \ \textit{such that}\ \ \mathbf{x}\le\mathbf{y}\ .$$
2. *The function $h:(\mathcal{R}^+)^n\to\mathcal{R}^+$ dominates the function $\tilde{h}:(\mathcal{R}^+)^n\to\mathcal{R}^+$ if there exists a constant η such that*
$$\tilde{h}(\mathbf{x})\le\eta h(\mathbf{x})\quad\forall\mathbf{x}\in(\mathcal{R}^+)^n\ .$$
3. *The function $h:(\mathcal{R}^+)^n\to\mathcal{R}^+$ is Lipschitz continuous if there exists a constant M such that*
$$|h(\mathbf{x})-h(\mathbf{y})|\le M\|\mathbf{x}-\mathbf{y}\|_\infty\quad\forall\mathbf{x},\mathbf{y}\in(\mathcal{R}^+)^n\ .$$

Assumption 1 *The function $f:(\mathcal{R}^+)^m\to\mathcal{R}^+$ is:*

1. *monotone;*
2. *linear with respect to scalar multiplications, i.e., $f(c\mathbf{x})=cf(\mathbf{x})$ for all $c\in\mathcal{R}^+$ and $\mathbf{x}\in(\mathcal{R}^+)^m$;*
3. *dominating the ℓ_∞ norm with a domination factor η_f that is independent of m;*
4. *Lipschitz continuous with a constant M_f that is independent of m;*
5. *computable by a recurrent formula (explained below).*

Assumption 2 *The functions $g^k : (\mathcal{R}^+)^d \to \mathcal{R}^+$ are:*

1. *dominating the ℓ_∞ and the ℓ_1 norms with a uniform domination factor η_g that is independent of m;*

2. *Lipschitz continuous with a uniform constant M_g that is independent of m.*

By assuming that f is computable by a recurrent formula we mean that there exists a family of functions $\psi^k(\cdot,\cdot)$, $1 \le k \le m$, such that

$$f(g^1,\ldots,g^k,0,\ldots,0) = \psi^k\left(f(g^1,\ldots,g^{k-1},0,\ldots,0),g^k\right) \tag{6}$$

(note that $f(0,\ldots,0)=0$ in view of Assumption 1-2). For example, if f is a weighted ℓ_p norm on $\mathcal{R}^m$, $1 \le p \le \infty$, with weights $(w_1,\ldots,w_m)$, then ψ^k is the ℓ_p norm on $\mathcal{R}^2$ with weights $(1,w_k)$.

Next, we see what functions comply with the above assumptions. Assumption 1 dictates a quite narrow class of outer cost functions. $f = \max$ is the most prominent member of that class (luckily, in many applications this is the only relevant choice of f). Other functions f for which our results apply are the ℓ_p norms taken on the t largest values in the argument vector, for some constant $t < m$; e.g., the sum of the two largest components. Assumption 1 is not satisfied by any of the usual ℓ_p norms for $p < \infty$ because of the conjunction of conditions 3 and 4: no matter how we rescale an ℓ_p norm, $p < \infty$, one of the parameters η_f (condition 3) or M_f (condition 4) would depend on m.

As for g^k, basically any norm on $\mathcal{R}^d$ is allowed. The most interesting choices are the ℓ_p norms and the Sobolev norms, $\|\boldsymbol{\ell}\|_{1,p} := \|\boldsymbol{\ell}\|_p + \|\boldsymbol{\Delta\ell}\|_p$ where $\boldsymbol{\Delta\ell} \in \mathcal{R}^{d-1}$ and $\boldsymbol{\Delta\ell}_j = \boldsymbol{\ell}_{j+1} - \boldsymbol{\ell}_j$, $1 \le j \le d-1$. Another natural choice is the "extensible bin" cost function,

$$g^k(\boldsymbol{\ell}^k) = \|\max\{\boldsymbol{\ell}^k, \mathbf{c}^k\}\| = \|(\max\{\boldsymbol{\ell}^k_1, \mathbf{c}^k_1\},\ldots,\max\{\boldsymbol{\ell}^k_d, \mathbf{c}^k_d\})\|; \tag{7}$$

here $\mathbf{c}^k$ is a constant vector reflecting the parameters of the kth machine and the outer norm may be any norm. In [8] we described a problem that arises in video transmission and broadcasting, the so called *line up problem*, where the above described choices for the inner cost functions are meaningful.

It is interesting to note that the set of functions that comply with either Assumption 1 or 2 is closed under positive linear combinations. For example, if f_1 and f_2 satisfy Assumption 1, so would $c_1 f_1 + c_2 f_2$ for all $c_1, c_2 > 0$.

We conclude this section by describing the difference between the above assumptions made on the cost functions, and the ones that we made in [8]. In [8], the outer cost function f had to comply with Assumption 1 above and, in addition, be symmetric with respect to its arguments and harmonic (A function $h : (\mathcal{R}^+)^m \to \mathcal{R}^+$ is harmonic if $h(\mathbf{x}) \ge h(\bar{\mathbf{x}})$ for all $\mathbf{x} \in (\mathcal{R}^+)^m$, where $\bar{\mathbf{x}} = (\bar{x},\ldots,\bar{x})$ and $\bar{x} = \frac{1}{m}\cdot\sum_{i=1}^m \mathbf{x}_i$). By removing these two properties, we are able to cover also weighted norms (such norms are neither symmetric nor harmonic).

As for the inner cost functions, the main improvement is that they may be different (compare (6) to (3)). Hence, the PTAS that we present here applies also to the case where the inner cost functions depend on different parameters, e.g., (11). A generalization of that sort is essential in the context of the line-up problem since different TV channels may be assigned different priorities, or may be associated with different mutiplexing hardware or software, and such differences would manifest themselves through different parameters in the inner cost functions. In addition, the assumptions that were made in [8] and are no longer required here are convexity and stable dependence with respect to scalar multiplications. The latter assumption means that $g(c\mathbf{x}) \leq \alpha(c)g(\mathbf{x})$ for all $c \in \mathcal{R}^+$ and $\mathbf{x} \in (\mathcal{R}^+)^d$, where $\alpha(c)$ is such that $\lg\lg\alpha(c)$ is bounded by a polynomial in c. That assumption was needed in order to bound the number of steps in the binary search that was implemented in [8] for the sake of approximating the optimal cost. Since the new method that we present here yields the required approximation in a one-time run and does not involve a binary search, we do not need to make this assumption on the inner cost functions. As it is hard to imagine reasonable cost functions that fail to satisfy this assumption, the removal of this assumption has mainly an aesthetic merit. Regarding the removal of the convexity assumption, this certainly extends the class of cost functions to which our analysis applies, since Assumption 2 does not imply convexity. For example, if $p : \mathcal{R}^+ \to \mathcal{R}^+$ is a non-convex function that dominates the identity function, then $g(\boldsymbol{\ell}) = \sum_{i=1}^d p(\boldsymbol{\ell}_i^k)$ is a function that complies with Assumption 2 above, but does not comply with Assumption 2 in [8].

We recall, see [8, Section 2], that the most suitable inner cost function for the line-up problem takes the form

$$g^k(\boldsymbol{\ell}^k) = \left\| \frac{(\boldsymbol{\ell}^k - c^k)_+}{\boldsymbol{\ell}^k} \right\| = \left\| \left(\frac{(\boldsymbol{\ell}_1^k - c^k)_+}{\boldsymbol{\ell}_1^k}, \ldots, \frac{(\boldsymbol{\ell}_d^k - c^k)_+}{\boldsymbol{\ell}_d^k} \right) \right\|$$

where c^k is a constant that stands for the bandwidth capacity of the kth channel while the outer norm may be any norm that best captures the requirements of the application (this cost function represents the reduction in the quality of the video at the output). This cost function is not convex, hence, the method in [8] does not apply to it. Regretfully, it does not comply with our Assumption 2-1 either; however, the method that we present here brings us one step closer to being able to cope with such cost functions as well.

3 A Graph Based Scheme

3.1 Preprocessing the vectors by means of truncation

Let I be the original instance of the VAP, (1). As in [8], we start by modifying I into another problem instance $\bar{I}$ where the vectors $\bar{\mathbf{x}}^i$ are defined by

$$\bar{\mathbf{x}}_j^i = \begin{cases} \mathbf{x}_j^i & \text{if } \mathbf{x}_j^i \geq \varepsilon \|\mathbf{x}^i\|_\infty \\ 0 & \text{otherwise} \end{cases} \qquad 1 \leq i \leq n \ , \ 1 \leq j \leq d \ . \tag{8}$$

The following lemma is an extension of [8, Lemma 1].

Lemma 1 *Let A be a solution to I and let $\bar{A}$ be the corresponding solution to $\bar{I}$. Then*

$$(1 - C_1\varepsilon)F(\bar{A}) \le F(A) \le (1 + C_1\varepsilon)F(\bar{A}) \quad \textit{where} \quad C_1 = M_g\eta_g \ . \tag{9}$$

Proof. Let $\boldsymbol{\ell}^k$ and $\bar{\boldsymbol{\ell}}^k$, $1 \le k \le m$, denote the load vectors in A and $\bar{A}$ respectively. In view of (12),

$$\bar{\boldsymbol{\ell}}^k \le \boldsymbol{\ell}^k \le \bar{\boldsymbol{\ell}}^k + \varepsilon \sum_{A(i)=k} \|\mathbf{x}^i\|_\infty \tag{10}$$

Since $\|\mathbf{x}^i\|_\infty = \|\bar{\mathbf{x}}^i\|_\infty \le \|\bar{\mathbf{x}}^i\|_1$ we conclude that $\sum_{A(i)=k} \|\mathbf{x}^i\|_\infty \le \|\bar{\boldsymbol{\ell}}^k\|_1$. Recalling Assumption 2-1 we get that

$$\sum_{A(i)=k} \|\mathbf{x}^i\|_\infty \le \eta_g g^k(\bar{\boldsymbol{\ell}}^k) \ . \tag{11}$$

Therefore, by (14) and (15),

$$\bar{\boldsymbol{\ell}}^k \le \boldsymbol{\ell}^k \le \bar{\boldsymbol{\ell}}^k + \varepsilon\eta_g g^k(\bar{\boldsymbol{\ell}}^k) \ .$$

Next, by the uniform Lipschitz continuity of g^k we conclude that

$$(1 - C_1\varepsilon)g^k(\bar{\boldsymbol{\ell}}^k) \le g^k(\boldsymbol{\ell}^k) \le (1 + C_1\varepsilon)g^k(\bar{\boldsymbol{\ell}}^k) \qquad \text{where} \ \ C_1 = M_g\eta_g \ .$$

Finally, we invoke the monotonicity of f and its linear dependence on scalar multiplications to arrive at (13). $\square$

We assume henceforth that the input vectors have been subjected to the truncation procedure (12). To avoid cumbersome notations we shall keep denoting the truncated vectors by $\mathbf{x}^i$ and their collection by I.

3.2 Large and small vectors

Let Φ^o denote the optimal cost, let A^o be an optimal solution, $F(A^o) = \Phi^o$, and let $\boldsymbol{\ell}^k$, $1 \le k \le m$, be the load vectors in that solution. Then, in view of Assumption 1-3 and Assumption 2-1,

$$\boldsymbol{\ell}^k \le \eta_f\eta_g\Phi^o \quad 1 \le k \le m \ .$$

Consequently, we conclude that all input vectors satisfy the same bound,

$$\mathbf{x}^i \le \eta_f\eta_g\Phi^o \quad 1 \le i \le n \ .$$

Hence, we get the following lower bound for the optimal cost:

$$\Phi^o \ge \Phi := \frac{\max_{1\le i\le n} \|\mathbf{x}^i\|_\infty}{\eta_f\eta_g} \ . \tag{12}$$

This lower bound induces a decomposition of the set of input vectors (1) into two subsets of large and small vectors as follows:

$$\mathcal{L} = \{\mathbf{x}^i \ : \|\mathbf{x}^i\|_\infty \geq \Phi\varepsilon^{2d+1}, \ 1 \leq i \leq n\} \ ,$$

$$\mathcal{S} = \{\mathbf{x}^i \ : \|\mathbf{x}^i\|_\infty < \Phi\varepsilon^{2d+1}, \ 1 \leq i \leq n\} \ .$$

We describe below a technique to replace $\mathcal{S}$ with another set of vectors $\tilde{\mathcal{S}} = \{\mathbf{z}^1, \dots, \mathbf{z}^{\tilde{\nu}}\}$ where

$$\tilde{\nu} = |\tilde{\mathcal{S}}| \leq \nu = |\mathcal{S}| \quad \text{and} \quad \|\mathbf{z}^i\|_\infty = \Phi\varepsilon^{2d+1} \ \ 1 \leq i \leq \tilde{\nu} \ . \tag{13}$$

In other words, all vectors in $\tilde{\mathcal{S}}$ are large.

We would like to stress that this technique was already introduced in [8] and we include its description here for the sake of completeness. It should be noted that in [8] the value of Φ was a test-value in the binary search that aims to approximate the optimal value. Hence, while in the PTAS described herein the technique that we proceed to describe is executed exactly once (with the value of Φ as in (20)), in [8] it is executed once in each iteration of the binary search.

Let $\mathbf{x} \in \mathcal{S}$. Then, in view of the truncation procedure (12),

$$\varepsilon \leq \frac{\mathbf{x}_j}{\|\mathbf{x}\|_\infty} \leq 1 \qquad \forall \mathbf{x}_j > 0 \ , \ 1 \leq j \leq d \ .$$

Next, we define a geometric mesh on the interval $[\varepsilon, 1]$:

$$\xi_0 = \varepsilon \ \ ; \quad \xi_i = (1+\varepsilon)\xi_{i-1} \ \ , \quad 1 \leq i \leq q \ \ ; \quad q := \left\lfloor \frac{-\lg \varepsilon}{\lg(1+\varepsilon)} \right\rfloor + 1 \ .$$

Every nonzero component of $\mathbf{x}/\|\mathbf{x}\|_\infty$ lies in an interval $[\xi_{i-1}, \xi_i)$ for some $1 \leq i \leq q$. Next, we define

$$\mathbf{x}' = \|\mathbf{x}\|_\infty \mathcal{H}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_\infty}\right) \ ,$$

where the operator $\mathcal{H}$ retains components that are 0 or 1 and replaces every other component by the left end point of the interval $[\xi_{i-1}, \xi_i)$ where it lies. Hence, since $\mathcal{H}$ may return one of the $q+2$ values $0, \xi_0, \dots, \xi_{q-1}, 1$, the vector $\mathbf{x}'$ may be in one of

$$s = (q+2)^d - 1$$

linear subspaces of dimension 1 in $\mathcal{R}^d$; we denote those subspaces by W^σ, $1 \leq \sigma \leq s$, and refer to them as *types*. In view of the above, we define the set

$$\mathcal{S}' = \{\mathbf{x}' \ : \ \ \mathbf{x} \in \mathcal{S}\} \ . \tag{14}$$

Next, we define for each type

$$\mathbf{w}^\sigma = \sum\{\mathbf{x}' \ : \ \ \mathbf{x}' \in \mathcal{S}' \cap W^\sigma\} \qquad 1 \leq \sigma \leq s \ ; \tag{15}$$

namely, $\mathbf{w}^\sigma$ aggregates all vectors $\mathbf{x}'$ of type σ. We now slice this vector into large identical "slices", where each of those slices and the number of slices are given by:

$$\tilde{\mathbf{w}}^\sigma = \frac{\mathbf{w}^\sigma}{\|\mathbf{w}^\sigma\|_\infty} \cdot \Phi\varepsilon^{2d+1} \quad \text{and} \quad \kappa_\sigma = \left\lceil \frac{\|\mathbf{w}^\sigma\|_\infty}{\Phi\varepsilon^{2d+1}} \right\rceil \ . \tag{16}$$

Finally, we define the set $\tilde{\mathcal{S}}$ as follows:

$$\tilde{\mathcal{S}} = \bigcup_{\sigma=1}^{s} \{\mathbf{z}^{\sigma,k} = \tilde{\mathbf{w}}^\sigma \ : \ 1 \le k \le \kappa_\sigma\} \ . \tag{17}$$

Namely, the new set $\tilde{\mathcal{S}}$ includes for each type σ the "slice"-vector $\tilde{\mathbf{w}}^\sigma$, (30), repeated κ_σ times. As implied by (30), the ℓ_∞ norm of all vectors in $\tilde{\mathcal{S}}$ is $\Phi\varepsilon^{2d+1}$, in accord with (23). Also, the number of vectors in $\tilde{\mathcal{S}}$, $\tilde{\nu} = \sum_{\sigma=1}^{s} \kappa_\sigma$, is obviously no more than ν as the construction of the new vectors implies that $\kappa_\sigma \le |\mathcal{S}' \cap W^\sigma|$ (recall that $\|\mathbf{x}'\|_\infty < \Phi\varepsilon^{2d+1}$ for all $\mathbf{x}' \in \mathcal{S}'$).

So we have modified the original problem instance I, having n input vectors $\mathcal{L} \cup \mathcal{S}$, into an intermediate problem instance $I' = \mathcal{L} \cup \mathcal{S}'$, see (28), and then to a new problem instance,

$$\tilde{I} = \mathcal{L} \cup \tilde{\mathcal{S}} \ , \tag{18}$$

see (29)-(31), that has $\tilde{n} = n - \nu + \tilde{\nu}$ input vectors. The following theorem states that those problem instances are close in the sense that for each solution of one problem instance there exists a solution of the other problem instance whose cost is almost the same. As it is a modified version of Theorem 3 in [8], we omit its proof.

Theorem 1 *For each solution $A \in \{1, \dots, m\}^{\{1,\dots,n\}}$ of I there exists a solution $\tilde{A} \in \{1, \dots, m\}^{\{1,\dots,\tilde{n}\}}$ of $\tilde{I}$ such that*

$$(1 - C_1\varepsilon) \cdot \left(F(\tilde{A}) - C_2\Phi\varepsilon\right) \le F(A) \le (1 + C_1\varepsilon) \cdot \left(F(\tilde{A}) + C_2\Phi\varepsilon\right) \ , \tag{19}$$

where C_1 is given in (13) and

$$C_2 = M_f M_g \ .$$

Conversely, for each solution $\tilde{A} \in \{1, \dots, m\}^{\{1,\dots,\tilde{n}\}}$ of $\tilde{I}$ there exists a solution $A \in \{1, \dots, m\}^{\{1,\dots,n\}}$ of I that satisfies (33).

3.3 The scheme

In view of the previous two subsections, we assume that the original set of input vectors I was subjected to the truncation procedure, along the lines of §3.1, and then modified into a problem instance $\tilde{I}$ where all vectors are large, using the procedure described in §3.2. For convenience, we shall keep denoting the number of input vectors in $\tilde{I}$ by n and the input vectors by $\mathbf{x}^i$, $1 \le i \le n$. Hence, all vectors in $\tilde{I}$ satisfy

$$\|\mathbf{x}^i\|_\infty \ge \Phi\varepsilon^{2d+1} \qquad 1 \le i \le n \ .$$

This, together with (12) on one hand and (20) on the other hand, yield the following lower and upper bounds:

$$\varepsilon^{2d+2} \le \frac{x_j^i}{\Phi} \le \eta_f \eta_g \qquad \text{for } \ 1 \le i \le n, \ 1 \le j \le d \ \text{ and } \ x_j^i \ne 0 \ . \tag{20}$$

Next, we define a geometric mesh on the interval given in (35):

$$\xi_0 = \varepsilon^{2d+2} \quad ; \quad \xi_i = (1+\varepsilon)\xi_{i-1} \ , \quad 1 \le i \le q \ ; \quad q := \left\lfloor \frac{\lg(\eta_f \eta_g \varepsilon^{-2(d+1)})}{\lg(1+\varepsilon)} \right\rfloor + 1 \ . \tag{21}$$

Every nonzero component of $\mathbf{x}^i/\Phi$, $1 \le i \le n$, lies in an interval $[\xi_{i-1}, \xi_i)$ for some $1 \le i \le q$. We use this in order to define a new set of vectors,

$$\hat{I} = \left\{ \hat{\mathbf{x}}^i = \Phi \mathcal{H}\left(\frac{\mathbf{x}^i}{\Phi}\right) \ : \ \mathbf{x}^i \in \tilde{I} \right\} \ , \tag{22}$$

where the operator $\mathcal{H}$ replaces each nonzero component in the vector on which it operates by the left end point of the interval $[\xi_{i-1}, \xi_i)$ where it lies.

Theorem 2 *Let $\tilde{A}$ be a solution of $\tilde{I}$ and let $\hat{A}$ be the corresponding solution of $\hat{I}$. Then*

$$(1 - C_1\varepsilon)F(\hat{A}) \le F(\tilde{A}) \le (1 + C_1\varepsilon)F(\hat{A}) \ , \tag{23}$$

where C_1 is given in (13).

Proof. Let $\tilde{\boldsymbol{\ell}}^k$ and $\hat{\boldsymbol{\ell}}^k$, $1 \le k \le m$, be the load vectors in $\tilde{A}$ and $\hat{A}$. Then

$$\hat{\boldsymbol{\ell}}^k \le \tilde{\boldsymbol{\ell}}^k \le (1+\varepsilon)\hat{\boldsymbol{\ell}}^k \qquad 1 \le k \le m \ .$$

Hence, $\|\tilde{\boldsymbol{\ell}}^k - \hat{\boldsymbol{\ell}}^k\|_\infty \le \varepsilon \|\hat{\boldsymbol{\ell}}^k\|_\infty$; using Assumptions 2-2 and 2-1 we get that

$$|g^k(\tilde{\boldsymbol{\ell}}^k) - g^k(\hat{\boldsymbol{\ell}}^k)| \le \varepsilon M_g \eta_g g^k(\hat{\boldsymbol{\ell}}^k) \qquad 1 \le k \le m \ ,$$

or, equivalently,

$$(1 - C_1\varepsilon)g^k(\hat{\boldsymbol{\ell}}^k) \le g^k(\tilde{\boldsymbol{\ell}}^k) \le (1 + C_1\varepsilon)g^k(\hat{\boldsymbol{\ell}}^k) \quad 1 \le k \le m \quad \text{where} \quad C_1 = M_g \eta_g \ .$$

These inequalities, together with the monotonicity of f and its linearity with respect to scalar multiplications (Assumptions 1-1 and 1-2) imply (38). □

The vectors in $\hat{I}$ belong to the set

$$W = \mathcal{X}^d \quad \text{where} \quad \mathcal{X} = \{0, \xi_0, \ldots, \xi_{q-1}\} \ .$$

As the size of W is $s = (q+1)^d$, it may be ordered:

$$W = \{\mathbf{w}^1, \ldots, \mathbf{w}^s\} \ . \tag{24}$$

With this, the set of modified vectors $\hat{I}$ may be identified by a configuration vector

$$\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_s) \quad \text{where} \quad \mathbf{z}_i = \#\{\hat{\mathbf{x}} \in \hat{I} \ : \ \hat{\mathbf{x}} = \mathbf{w}^i\} \ , \ 1 \le i \le s \ . \tag{25}$$

Next, we may describe all possible assignments of vectors from $\hat{I}$ to the m machines using a layered graph $G = (V, E)$. To that end, assume that $\hat{A} : \hat{I} \to \{1, \ldots, m\}$ is one of these assignments. We let $\hat{I}^k$ denote the subset of $\hat{I}$ consisting of those vectors that were assigned to one of the first k machines,

$$\hat{I}^k = \{\hat{\mathbf{x}} \in \hat{I} \ : \ \hat{A}(\hat{\mathbf{x}}) \le k\} \qquad 1 \le k \le m \ .$$

Furthermore, we define the corresponding *state vector*

$$\mathbf{z}^k = (\mathbf{z}_1^k, \ldots, \mathbf{z}_s^k) \quad 1 \le k \le m \quad \text{where} \quad \mathbf{z}_i^k = \#\{\hat{\mathbf{x}} \in \hat{I}^k \ : \ \hat{\mathbf{x}} = \mathbf{w}^i\} \ , \ 1 \le i \le s \ .$$

We note that

$$\emptyset = \hat{I}^0 \subseteq \hat{I}^1 \subseteq \ldots \subseteq \hat{I}^{m-1} \subseteq \hat{I}^m = \hat{I}$$

and

$$\mathbf{0} = \mathbf{z}^0 \le \mathbf{z}^1 \le \ldots \le \mathbf{z}^{m-1} \le \mathbf{z}^m = \mathbf{z} \ ,$$

where $\mathbf{z}$ is given in (41). In addition, when $0 < k < m$, $\hat{I}^k$ may be any subset of $\hat{I}$ while $\mathbf{z}^k$ may be any vector in

$$Z = \{\mathbf{y} \in \mathbb{Z}^s \ : \ \mathbf{0} \le \mathbf{y} \le \mathbf{z}\} \ . \tag{26}$$

With this, we define the graph $G = (V, E)$ as follows:

- The set of vertices consists of $m + 1$ layers, $V = \bigcup_{k=0}^m V^k$. If $v \in V$ is a vertex in the kth layer, V^k, then it represents one of the possible state vectors after assigning vectors to the first k machines. Hence $V^0 = \{\mathbf{0}\}$, $V^m = \{\mathbf{z}\}$ and the intermediate layers are $V^k = Z$, see (44), $0 < k < m$.

- The set of edges consists of m subsets:

  $$E = \bigcup_{k=1}^m E^k \quad \text{where} \quad E^k = \{(\mathbf{u}, \mathbf{v}) \ : \ \mathbf{u} \in V^{k-1} \ , \ \mathbf{v} \in V^k \ , \ \mathbf{u} \le \mathbf{v}\} \ .$$

  In other words, there is an edge connecting two vertices in adjacent layers, $\mathbf{u} \in V^{k-1}$ and $\mathbf{v} \in V^k$, if and only if there exists an assignment to the kth machine that would change the state vector from $\mathbf{u}$ to $\mathbf{v}$.

Note that all intermediate layers, V^k, $0 < k < m$, are composed of the same number of vertices, t, given by the number of sub-vectors that $\mathbf{z}$ has:

$$t = |Z| = \prod_{i=1}^s (\mathbf{z}_i + 1) \le (n + 1)^s \ .$$

Next, we turn the graph into a weighted graph, using a weight function $w : E \to \mathcal{R}^+$ that computes the cost that the given edge implies on the corresponding machine: Let $e = (\mathbf{u}, \mathbf{v}) \in E^k$. Then the difference $\mathbf{v} - \mathbf{u}$ tells us how many vectors of each of the s types are assigned by this edge to the kth machine. The weight of this edge is therefore defined as

$$w(e) = g^k(T(\mathbf{v} - \mathbf{u})) \quad \text{where} \quad T(\mathbf{v} - \mathbf{u}) = \sum_{i=1}^{s} (\mathbf{v}_i - \mathbf{u}_i)\mathbf{w}^i \ ,$$

$\mathbf{w}^i$ are as in (40). We define a cost function on the vertices, $r : V \to \mathcal{R}^+$; this cost function is defined recursively using Assumption 1-5:

$$r(v) = 0 \quad , \quad v \in V^0 \ ;$$

$$r(v) = \min \left\{ \psi^k(r(u), w(e)) \ : \ u \in V^{k-1}, \ e = (u, v) \in E^k \right\} \quad , \quad v \in V^k$$

(the functions ψ^k are as in (10)). This cost function coincides with the cost function of the VAP, (6). More specifically, if $v \in V^k$ and it represents a subset of vectors $\hat{I}^k \subseteq \hat{I}$, then $r(v)$ equals the value of an optimal assignment of the vectors in $\hat{I}^k$ to the first k machines. Hence, the cost of the end vertex, $r(v)$, $v \in V^m$, equals the value of an optimal solution of the VAP for $\hat{I}$.

The goal is to find the shortest path from V^0 to V^m that achieves this minimal cost. Namely, we look for a sequence of vertices $v^k \in V^k$, $0 \le k \le m$, such that

$$e^k := (v^{k-1}, v^k) \in E^k \qquad 1 \le k \le m$$

and

$$f(w(e^1), \ldots, w(e^m)) = r(v^m) \ .$$

We may apply a standard algorithm to find this minimal path within $\mathcal{O}(|V| + |E|)$ steps. As

$$|V| \le 2 + (m-1) \cdot (n+1)^s \tag{27}$$

and

$$|E| = \sum_{k=1}^{m} |E^k| \le m \cdot (n+1)^{2s} \ , \tag{28}$$

where $s = (q+1)^d$ and q depends only on d, f, g and ε, see (36), the running time would be polynomial in n and m.

The shortest path thus found represents an assignment of the vectors of the modified set $\hat{I}$,

$$\hat{A} : \hat{I} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\} \to \{1, \ldots, m\} \ .$$

We need to translate this assignment into an assignment of the original vectors,

$$A : I = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \to \{1, \ldots, m\} \ .$$

To that end, let us review all the problem modifications that we performed:

- First modification: I to $\bar{I}$, see (12) and Lemma 1.
- Second modification: $\bar{I}$ to $\tilde{I}$, see (29)-(32) and Theorem 1.
- Third modification: $\tilde{I}$ to $\hat{I}$, see (37) and Theorem 2.

In view of the above, we translate the solution that we found, $\hat{A}$, into a solution $\tilde{A}$ of $\tilde{I}$, then – along the lines of Theorem 1 – we translate it into a solution $\bar{A}$ of $\bar{I}$ and finally we take the corresponding solution A of I.

Theorem 3 *Let Φ^o be the optimal cost of the original problem instance I. Let A be the solution of I that is obtained using the above scheme. Then A satisfies (5) with a constant that depends only on η_g, M_g and M_f.*

Proof. Let $\hat{\Phi}^o$ denote the optimal cost of $\hat{I}$. Then, using the left inequalities in Lemma 1, Theorem 1 and Theorem 2, together with (20), we conclude that

$$\hat{\Phi}^o \le \frac{1}{1-C_1\varepsilon} \cdot \left(\frac{1}{(1-C_1\varepsilon)^2} + C_2\varepsilon \right) \Phi^o \le (1+T\varepsilon)\Phi^o \ , \tag{29}$$

for an appropriate choice of T that depends only on C_1 and C_2. However, $\hat{\Phi}^o$ is no other than the cost of the shortest path that we found in the graph, namely, the cost of the solution $\hat{A}$ that we found for $\hat{I}$. As A is the solution of I that is obtained from $\hat{A}$, we may upper bound its cost using the right inequalities in Lemma 1, Theorem 1 and Theorem 2:

$$F(A) \le (1+C_1\varepsilon)^2 \cdot \left((1+C_1\varepsilon)\hat{\Phi}^o + C_2\Phi^o\varepsilon \right) \ . \tag{30}$$

The Inequalities (56) and (57) imply that

$$F(A) \le (1+C_1\varepsilon)^2 \cdot ((1+C_1\varepsilon)(1+T\varepsilon) + C_2\varepsilon)\, \Phi^o \le (1+C\varepsilon)\Phi^o \ ,$$

where the constant C depends only on $C_1 = M_g\eta_g$ and $C_2 = M_f M_g$. $\Box$

The complexity of the preprocessing steps, i.e., the computation of the configuration vector $\mathbf{z}$, (41), that corresponds to the input vectors I, is $\mathcal{O}(nd)$. This is also the complexity of the a-posteriori steps that aim to recover an assignment of the original vectors that corresponds to the minimal path in the graph. Hence, the overall complexity of our scheme is determined by the complexity of the algorithm to find the shortest path, which equals

$$\mathcal{O}(|V|+|E|) = \mathcal{O}(|E|) = \mathcal{O}(m\cdot(n+1)^{2s}) \quad \text{where} \quad s = \left(\left\lfloor \frac{\lg(\eta_f\eta_g\varepsilon^{-2(d+1)})}{\lg(1+\varepsilon)} \right\rfloor + 2 \right)^d \ ,$$

see (52), (53) and (36).

References

[1] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms (SODA'97)*, pages 493–500, 1997.

[2] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, Vol. 1, pages 55–66, 1998.

[3] Y. Azar and L. Epstein. Approximation schemes for covering and scheduling on related machines. In *1st Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX98)*, pages 39–47, 1998.

[4] C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 185–194, 1999.

[5] E. G. Coffman, Jr. and G. S. Lueker. Approximation algorithms for extensible bin packing. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 586–588, 2001.

[6] P. Dell'Olmo, H. Kellerer, M. G. Speranza, and Zs. Tuza. A 13/12 approximation algorithm for bin packing with extendable bins. *Information Processing Letters*, Vol. 65, pages 229–233, 1998.

[7] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *Proceedings of the 7th Annual European Symposium on Algorithms (ESA'99)*, pages 151–162, 1999.

[8] L. Epstein and T. Tassa. Vector assignment problems: A general framework. *Journal of Algorithms*, Vol. 48, pages 360–384, 2003.

[9] L. Epstein and T. Tassa. Vector assignment problems: A general framework. *Proceedings of the 10th Annual European Symposium on Algorithms (ESA 2002)*, pages 461-472, 2002.

[10] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, Vol. 45, pages 1563–1581, 1966.

[11] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, Vol. 17, pages 539–551, 1988.

[12] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, Vol. 34, pages 144–162, 1987.