Vector Assignment Problems: A General Framework ⋆

Leah Epstein[1] and Tamir Tassa[2]

[1] School of Computer Science, The Interdisciplinary Center, P.O.B 167, 46150 Herzliya, Israel. lea@idc.ac.il.

[2] Department of Applied Mathematics, Tel-Aviv University, Ramat Aviv, Tel Aviv, Israel. tassa@post.tau.ac.il.

Abstract. We present a general framework for vector assignment problems. In such problems one aims at assigning n input vectors to m machines such that the value of a given target function is minimized. While previous approaches concentrated on simple target functions such as max-max, the general approach presented here enables us to design a PTAS for a wide class of target functions. In particular we are able to deal with non-monotone target functions and asymmetric settings where the cost functions per machine may be different for different machines. This is done by combining a graph-based technique and a new technique of preprocessing the input vectors.

1 Introduction

In this paper we present a general framework for dealing with assignment problems in general and vector assignment problems in particular. An assignment problem is composed of the following three ingredients:

- Items: $x^1, \ldots, x^n$;
- Containers: $c^1, \ldots, c^m$;
- A target function: $F : \{1, \ldots, m\}^{\{1,\ldots,n\}} \to \mathcal{R}^+$.

Each item is characterized by a parameter or a set of parameters that reflect the "size" of the item. That size may be a scalar, a vector or whatever the application which gave rise to the problem dictates.
The containers may be characterized by their capacity; that capacity would be a scalar or a vector, in accord with the type of the items to be stored.
The set $\{1, \ldots, m\}^{\{1,\ldots,n\}}$ consists of all possible assignments of items to containers. Each assignment is referred to as *a solution* to the problem. In all assignment problems there is a natural addition operation between items. Hence, given an assignment (solution) $A \in \{1, \ldots, m\}^{\{1,\ldots,n\}}$, we may compute the *load* in each container as

$$l^k = \sum_{A(i)=k} x^i \ .$$

⋆ Research supported in part by the Israel Science Foundation (grant no. 250/01).

The target function evaluates for each solution a nonnegative *cost*. That function takes into account the loads l^k and possibly also the container capacities, if such capacities are given.

Such problems are known to be strongly NP-hard. Hence, polynomial time approximation schemes (PTAS) are sought. Such schemes produce, in polynomial time, a solution (i.e., an assignment) whose cost is larger than that of an optimal solution by a factor of no more than $(1 + \text{Const} \cdot \varepsilon)$, where $\varepsilon > 0$ is an arbitrary parameter. Namely, if Φ^o is the optimal cost and $\varepsilon > 0$ is a given parameter, the scheme produces a solution A that satisfies

$$F(A) \leq (1 + \text{Const} \cdot \varepsilon) \cdot \Phi^o \ , \tag{1}$$

where the constant is independent of the input data (n, m and $\mathbf{x}^i$, $1 \leq i \leq n$) and ε, but may depend on the dimension of the vectors.

The above formulation encompasses all problems that were studied in the art so far. However, the chosen target functions in those studies were limited to a narrow class of "natural" functions, as described below. Motivated by an interesting problem that arises in transmitting multiplexed video streams, we suggest here a general framework that includes a much wider class of target functions.

Overview. We focus here on Vector Assignment Problems (VAP), where the items $\mathbf{x}^i$, $1 \leq i \leq n$, and the resulting loads $\mathbf{l}^k$, $1 \leq k \leq m$, are vectors in $(\mathcal{R}^+)^d$. We consider target functions of the form:

$$F(A) = f(g^1(\mathbf{l}^1), \ldots, g^m(\mathbf{l}^m)) \ . \tag{2}$$

Here: A is a given solution and $\mathbf{l}^k$, $1 \leq k \leq m$, are the corresponding load vectors. $g^k : (\mathcal{R}^+)^d \to \mathcal{R}^+$ $(1 \leq k \leq m)$ are functions that evaluate a cost per container. $f : (\mathcal{R}^+)^m \to \mathcal{R}^+$ is a function that evaluates the final cost over all containers.

Relation to previously studied problems. This suggested framework includes many problems that are already known in the art. The terminology in those problems may vary. In some scalar problems the containers are referred to as *bins*. In other scalar problems and in most all vector problems the terms *items*, *containers* and *assignment* are replaced with *jobs*, *machines* and *scheduling*, respectively. Since we have in mind applications that do not deal with scheduling, we adopt herein a slightly more general terminology: *vectors, machines* and *assignment*.

Herein, we list some of those problems. The first 4 examples are scalar. The last one is a vector problem.

1. The classical problem in this context is the scalar *makespan problem*. In that problem one aims at minimizing the maximal load. It is described by (2) with $d = 1$, $g^k = id$ and $f = \max$. See [11–13, 15].
2. The ℓ_p minimization problem is given by (2) with $d = 1$, $g^k(x) = x^p$ and $f(y_1, \ldots, y_m) = \sum_{k=1}^{m} y_k^p$. The case $p = 2$ was studied in [2, 4] and was motivated by storage allocation problems. The general case was studied in [1].

3. Problem (2) with $d = 1$, $g^k(x) = h(x)$ for all k, where $h : \mathcal{R}^+ \to \mathcal{R}^+$ is some fixed function, and f is either the maximum or sum of its arguments, was studied in [1]. Other choices for f are the inverse minimum or the inverse sum. By considering those choices, one aims at **maximizing** the minimal or average completion time. See also [6, 16].
4. The Extensible Bin Packing Problem is given by (2) with $d = 1$, $g^k(x) = \max\{x, 1\}$ for all k and $f(y_1, \ldots, y_m) = \sum_{k=1}^{m} y^k$. See [5, 8, 9].
5. The Vector Scheduling Problem, see [3], coincides with (2) with $f = g^k = \max$.

In most of the above examples the target functions were monotone. Namely, when adding an item to a container, the value of the target function increases, or at least does not decrease. Such monotonicity is indeed natural when dealing with bin packing or job scheduling: every item that is stored in a bin decreases the remaining available space in that bin; every job assigned to a machine increases the load on that machine. However, we present in this paper the so called *line-up problem* that arises in video transmission and broadcasting, where the target function has a different nature: it aims at optimizing the quality of the transmitted video. Such functions are not monotone - increasing the size of a vector component may actually decrease the value of the target function.

We note in passing that a related class of problems that we exclude from our discussion is that in which the goal is to minimize the number of containers that are used for packing, subject to some condition (such conditions are usually associated with the capacity of the containers). See [10, 7, 14, 17, 3].

Notation agreements. Throughout this paper we adopt the following conventions:

– 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., $\mathbf{l}^k_j$ denotes the jth component of the vector $\mathbf{l}^k$.
– 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 x is a scalar then $x_+ = \max\{x, 0\}$.
– If $\circ$ is any operation between scalars then $\mathbf{v} \circ c$ is the vector whose jth component (for all values of j) is $\mathbf{v}_j \circ c$. Similarly, if $\propto$ is any relation between scalars, then $\mathbf{v} \propto c$ or $\mathbf{v} \propto \mathbf{w}$ mean that the relation holds component-wise.

2 The Cost Functions

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

Definition 1.

1. *A function $h : (\mathcal{R}^+)^n \to \mathcal{R}^+$ is monotone if*

$$h(\mathbf{x}) \le h(\mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y} \in (\mathcal{R}^+)^n \ \text{ such that } \ \mathbf{x} \le \mathbf{y} \ . \tag{3}$$

2. *The function $h : (\mathcal{R}^+)^n \to \mathcal{R}^+$ is dominating the function $\tilde{h} : (\mathcal{R}^+)^n \to \mathcal{R}^+$ if there exists a constant η such that*

$$\tilde{h}(\mathbf{x}) \le \eta h(\mathbf{x}) \quad \forall \mathbf{x} \in (\mathcal{R}^+)^n \ . \tag{4}$$

3. *The function $h : (\mathcal{R}^+)^n \to \mathcal{R}^+$ is Lipschitz continuous if there exists a constant M such that*

$$|h(\mathbf{x}) - h(\mathbf{y})| \le M \|\mathbf{x} - \mathbf{y}\|_\infty \quad \forall \mathbf{x}, \mathbf{y} \in (\mathcal{R}^+)^n \ . \tag{5}$$

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 max 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. *recursively computable (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 domination factor η_g that does not depend on m;*
2. *Lipschitz continuous with a constant M_g that does not depend on m.*

By assuming that f is recursively computable 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, where $t = \min(m_0, m)$ for some constant m_0; e.g., the sum of the largest two 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, $\|\mathbf{l}\|_{1,p} := \|\mathbf{l}\|_p + \|\mathbf{\Delta l}\|_p$ where $\mathbf{\Delta l} \in \mathcal{R}^{d-1}$ and $\mathbf{\Delta l}_j = \mathbf{l}_{j+1} - \mathbf{l}_j$, $1 \le j \le d-1$. Another natural choice is the "extensible bin" cost function, $g^k(\mathbf{l}^k) = \|\max\{\mathbf{l}^k, \mathbf{c}^k\}\|$; here $\mathbf{c}^k$ is a constant vector reflecting the parameters of the kth machine and the outer norm may be any norm.

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 $c_1, c_2 > 0$.

3 A Graph Based Scheme

3.1 Preprocessing the Vectors by Means of Truncation

Let I be the original instance of the VAP. We start by modifying I into another problem instance $\bar{I}$ where the vectors $\bar{\mathbf{x}}^i$ are defined by

$$\bar{\mathbf{x}}^i_j = \begin{cases} \mathbf{x}^i_j & \text{if } \ \mathbf{x}^i_j \ge \varepsilon \|\mathbf{x}^i\|_\infty \\ 0 & \text{otherwise} \end{cases} \qquad 1 \le i \le n \ , \ 1 \le j \le d \ . \tag{7}$$

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

$$(1 - C_1\varepsilon)F(\bar{A}) \le F(A) \le (1 + C_1\varepsilon)F(\bar{A}) \quad \textit{where} \quad C_1 = M_g \eta_g \ . \tag{8}$$

Proof. Let $\mathbf{l}^k$ and $\bar{\mathbf{l}}^k$, $1 \le k \le m$, denote the load vectors in A and $\bar{A}$ respectively. In view of (7),

$$\bar{\mathbf{l}}^k \le \mathbf{l}^k \le \bar{\mathbf{l}}^k + \varepsilon \sum_{A(i)=k} \|\mathbf{x}^i\|_\infty \tag{9}$$

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

$$\sum_{A(i)=k} \|\mathbf{x}^i\|_\infty \le \eta_g g^k(\bar{\mathbf{l}}^k) \ . \tag{10}$$

Therefore, by (9) and (10), $\bar{\mathbf{l}}^k \le \mathbf{l}^k \le \bar{\mathbf{l}}^k + \varepsilon \eta_g g^k(\bar{\mathbf{l}}^k)$. Next, by the uniform Lipschitz continuity of g^k we conclude that

$$(1 - C_1\varepsilon)g^k(\bar{\mathbf{l}}^k) \le g^k(\mathbf{l}^k) \le (1 + C_1\varepsilon)g^k(\bar{\mathbf{l}}^k) \qquad \text{where} \ \ C_1 = M_g \eta_g \ . \tag{11}$$

Finally, we invoke the monotonicity of f and its linear dependence on scalar multiplications to arrive at (8). $\square$

We assume henceforth that the input vectors have been subjected to the truncation procedure (7). 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 $\mathbf{l}^k$, $1 \le k \le m$, be the load vectors in that solution. Then, in view of Assumption 1-3 and Assumption 2-1, $\mathbf{l}^k \le \eta_f \eta_g \Phi^o, \quad 1 \le k \le m$. 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 into two subsets (multi-sets) of large and small vectors as follows:

$$\mathcal{L} = \{\mathbf{x}^i \ : \|\mathbf{x}^i\|_\infty \ge \Phi \varepsilon^{2d+1}, \ 1 \le i \le n\} \ , \tag{13}$$

$$\mathcal{S} = \{\mathbf{x}^i \ : \|\mathbf{x}^i\|_\infty < \Phi \varepsilon^{2d+1}, \ 1 \le i \le n\} \ . \tag{14}$$

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

$$\tilde{\nu} = |\tilde{\mathcal{S}}| \le \nu = |\mathcal{S}| \quad \text{and} \quad \|\mathbf{z}^i\|_\infty = \Phi \varepsilon^{2d+1} \ \ 1 \le i \le \tilde{\nu} \ . \tag{15}$$

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

Let $\mathbf{x} \in \mathcal{S}$. Then, in view of the truncation procedure (7),

$$\varepsilon \le \frac{\mathbf{x}_j}{\|\mathbf{x}\|_\infty} \le 1 \qquad \forall \mathbf{x}_j > 0 \ , \ 1 \le j \le d \ . \tag{16}$$

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

$$\xi_0 = \varepsilon \ \ ; \quad \xi_i = (1+\varepsilon)\xi_{i-1} \ \ , \quad 1 \le i \le q \ \ ; \quad q := \left\lfloor \frac{-\lg \varepsilon}{\lg(1+\varepsilon)} \right\rfloor + 1 \ . \tag{17}$$

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

$$\mathbf{x}' = \|\mathbf{x}\|_\infty \mathcal{H}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_\infty}\right) \ , \tag{18}$$

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, 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, $1 \le \sigma \le s$. In view of the above, we define the set

$$\mathcal{S}' = \{\mathbf{x}' \ : \ \ \mathbf{x} \in \mathcal{S}\} \ . \tag{19}$$

Next, we define for each type $1 \le \sigma \le s$

$$\mathbf{w}^\sigma = \sum \{\mathbf{x}' \ : \ \ \mathbf{x}' \in \mathcal{S}' \cap W^\sigma\} \ ; \tag{20}$$

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 their number are given by:

$$\tilde{\mathbf{w}}^\sigma = \frac{\mathbf{w}^\sigma}{\|\mathbf{w}^\sigma\|_\infty} \cdot \Phi\varepsilon^{2d+1} \quad \text{and} \quad \kappa_\sigma = \left\lceil \frac{\|\mathbf{w}^\sigma\|_\infty}{\Phi\varepsilon^{2d+1}} \right\rceil \ . \tag{21}$$

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

$$\tilde{\mathcal{S}} = \cup_{\sigma=1}^{s} \{\mathbf{z}^{\sigma,q} = \tilde{\mathbf{w}}^\sigma \ : \ 1 \le q \le \kappa_\sigma\} \ . \tag{22}$$

Namely, the new set $\tilde{\mathcal{S}}$ includes for each type σ the "slice"-vector $\tilde{\mathbf{w}}^\sigma$, (21), repeated κ_σ times. As implied by (21), all vectors in $\tilde{\mathcal{S}}$ have a max norm of $\Phi\varepsilon^{2d+1}$, in accord with (15). 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 (19), and then to a new problem instance,

$$\tilde{I} = \mathcal{L} \cup \tilde{\mathcal{S}} \ , \tag{23}$$

see (20)-(22), that has $\tilde{n} = n - \nu + \tilde{\nu}$ input vectors. The following theorem states that those problem instances are close in the sense that interests us.

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

$$(1 - C_1\varepsilon) \cdot \left(F(\tilde{A}) - C_2\Phi\varepsilon\right) \le F(A) \le (1 + C_1\varepsilon) \cdot \left(F(\tilde{A}) + C_2\Phi\varepsilon\right) \ , \tag{24}$$

where C_1 is given in (8) and

$$C_2 = M_f M_g \ . \tag{25}$$

Conversely, for each solution $\tilde{A} \in \{1, \ldots, m\}^{\{1,\ldots,\tilde{n}\}}$ of $\tilde{I}$ there exists a solution $A \in \{1, \ldots, m\}^{\{1,\ldots,n\}}$ of I that satisfies (24).

Proof. Let A be a solution of I and A' be its counterpart solution of I'. Let $\mathbf{l}^k$ and $\mathbf{l}'^k$, $1 \le k \le m$, denote the load vectors in A and A', respectively. By (18), $1 \le \mathbf{l}^k / \mathbf{l}'^k \le 1 + \varepsilon$. Hence, by Assumption 2-1,

$$\|\mathbf{l}^k - \mathbf{l}'^k\|_\infty \le \varepsilon \eta_g g^k(\mathbf{l}'^k) \ . \tag{26}$$

Therefore, by the uniform Lipschitz continuity of g^k,

$$(1 - C_1\varepsilon) g^k(\mathbf{l}'^k) \le g^k(\mathbf{l}^k) \le (1 + C_1\varepsilon) g^k(\mathbf{l}'^k) \quad 1 \le k \le m \tag{27}$$

where C_1 is as in (8). Applying f on (27) and using Assumptions 1-1 and 1-2, we get that

$$(1 - C_1\varepsilon) F(A') \le F(A) \le (1 + C_1\varepsilon) F(A') \ . \tag{28}$$

Next, given a solution A' of I', we construct a solution $\tilde{A}$ of $\tilde{I}$ such that

$$F(\tilde{A}) - C_2 \Phi \varepsilon \leq F(A') \leq F(\tilde{A}) + C_2 \Phi \varepsilon \ , \tag{29}$$

with C_2 as in (25). Showing this will enable us to construct for any solution A of I a solution $\tilde{A}$ of $\tilde{I}$ for which, in view of (28) and (29), (24) holds. Then, in order to complete the proof, we shall show how from a given solution $\tilde{A}$ of $\tilde{I}$, we are able to construct a solution A' of I' for which (29) holds.

To this end, we fix $1 \leq \sigma \leq s$ and define for every machine k the following vector:

$$\mathbf{y}^{\sigma,k} = \sum \{ \mathbf{x}'^i \ : \ \mathbf{x}'^i \in \mathcal{S}' \cap W^\sigma \ , \ A'(i) = k \} \ ; \tag{30}$$

i.e., $\mathbf{y}^{\sigma,k}$ is the sum of small vectors of type σ in I' that are assigned to the kth machine. Recalling (21), $\tilde{\mathcal{S}}$ includes the vector $\tilde{\mathbf{w}}^\sigma$ repeated κ_σ times, where

$$\kappa_\sigma = \left\lceil \sum_{k=1}^{m} \frac{\|\mathbf{y}^{\sigma,k}\|_\infty}{\Phi \varepsilon^{2d+1}} \right\rceil \ . \tag{31}$$

We may now select for each k an integer $t_{\sigma,k}$ such that

$$\left| t_{\sigma,k} - \frac{\|\mathbf{y}^{\sigma,k}\|_\infty}{\Phi \varepsilon^{2d+1}} \right| \leq 1 \tag{32}$$

and $\sum\limits_{k=1}^{m} t_{\sigma,k} = \kappa_\sigma$. The integers $t_{\sigma,k}$ can be found in the following manner:

We define $t_{\sigma,k}^{low} = \lfloor \|\mathbf{y}^{\sigma,k}\|_\infty / \Phi \varepsilon^{2d+1} \rfloor$ and $t_{\sigma,k}^{high} = \lceil \|\mathbf{y}^{\sigma,k}\|_\infty / \Phi \varepsilon^{2d+1} \rceil$. Clearly, $\sum\limits_{k=1}^{m} t_{\sigma,k}^{low} \leq \kappa_\sigma$ and $\sum\limits_{k=1}^{m} t_{\sigma,k}^{high} \geq \kappa_\sigma$. Since $t_{\sigma,k}^{high} - t_{\sigma,k}^{low} \leq 1$ for all $1 \leq k \leq m$, there exists an integer number $0 \leq x \leq m$ such that $\sum_{k=1}^{m} t_{\sigma,k}^{low} = \kappa_\sigma - x$. Finally, we set

$$t_{\sigma,k} = \begin{cases} t_{\sigma,k}^{high} & 1 \leq k \leq x \\ \\ t_{\sigma,k}^{low} & x < k \leq m \end{cases}$$

With this, the solution $\tilde{A}$ is that which assigns to the kth machine, $1 \leq k \leq m$, $t_{\sigma,k}$ vectors $\tilde{\mathbf{w}}^\sigma$ for all $1 \leq \sigma \leq s$ (and coincides with A' for all large vectors in $\mathcal{L}$). In view of (32) and the definition of $\tilde{\mathbf{w}}^\sigma$, see (21),

$$\|t_{\sigma,k} \cdot \tilde{\mathbf{w}}^\sigma - \mathbf{y}^{\sigma,k}\|_\infty \leq \Phi \varepsilon^{2d+1} \ . \tag{33}$$

Therefore, summing (33) over $1 \leq \sigma \leq s$, we conclude that $\tilde{\mathbf{l}}^k$ and $\mathbf{l}'^k$ – the loads on the kth machine in $\tilde{A}$ and A' respectively – are close,

$$\|\tilde{\mathbf{l}}^k - \mathbf{l}'^k\|_\infty \leq s \Phi \varepsilon^{2d+1} \ . \tag{34}$$

However, as (17) and the definition of s imply that $s \leq \varepsilon^{-2d}$ for all $0 < \varepsilon \leq 1$. We conclude by that $\|\tilde{\mathbf{l}}^k - \mathbf{l}'^k\|_\infty \leq \Phi \varepsilon$. Finally, the Lipschitz continuity of both g and f imply that (29) holds with C_2 as in (25).

Next, we show how to construct from a solution $\tilde{A}$ of $\tilde{I}$, a solution A' of I' for which (29) holds. The two assignments will coincide for the large vectors $\mathcal{L}$. As for the small vectors, let us fix one vector type $1 \le \sigma \le s$, where s is the number of types. $\tilde{\mathcal{S}}$ includes the vector $\tilde{\mathbf{w}}^\sigma$ repeated κ_σ times, (21)-(22). Let $t_{\sigma,k}$ be the number of those vectors that $\tilde{A}$ assigns to the kth machine. The counters $t_{\sigma,k}$ satisfy the bounds on them. We now assign the vectors $\mathbf{x}' \in \mathcal{S}' \cap W^\sigma$, see (20), to the m machines so that the ℓ_∞-norm of their sum in the kth machine is greater than $(t_{\sigma,k} - 1)\Phi\varepsilon^{2d+1}$ but no more than $(t_{\sigma,k} + 1)\Phi\varepsilon^{2d+1}$. In view of (20) and (21), it is easy to see that such an assignment exists: Assign the jobs one by one greedily, in order to obtain in the kth machine, $1 \le k \le m$, a load with an ℓ_∞-norm of at least $(t_{\sigma,k} - 1)\Phi\varepsilon^{2d+1}$. Since the ℓ_∞-norm of the sum of all small jobs is at least $(\sum_{k=1}^m t_{\sigma,k} - 1)\Phi\varepsilon^{2d+1}$, this goal can be achieved. Also, as the size of each of those jobs is no more than $\Phi\varepsilon^{2d+1}$, we may perform this assignment in a manner that keeps the load in each machine below $t_{\sigma,k}\Phi\varepsilon^{2d+1}$. After achieving that goal in all machines, we assign the remaining jobs so that the total load in each machine is bounded by $(t_{\sigma,k} + 1)\Phi\varepsilon^{2d+1}$. This is possible given the small size of the jobs and the size of their sum (at most $(\sum_{k=1}^m t_{\sigma,k})\Phi\varepsilon^{2d+1}$). Clearly, if we let $\mathbf{y}^{\sigma,k}$ denote the sum of vectors $\mathbf{x}'$ of type σ thus assigned to the kth machine, then $\mathbf{y}^{\sigma,k}$ satisfies (33). As we saw before, this implies that $\tilde{A}$ and A' satisfy (29). This completes the proof.□

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 $\|\mathbf{x}^i\|_\infty \ge \Phi\varepsilon^{2d+1}$ for $1 \le i \le n$. This, together with (7) on one hand and (12) on the other hand, yield the following lower and upper bounds:

$$\varepsilon^{2d+2} \le \frac{x_j^i}{\Phi} \le \eta_f \eta_g \qquad \text{for} \;\; 1 \le i \le n,\; 1 \le j \le d \;\; \text{and} \;\; x_j^i \ne 0 \; . \tag{35}$$

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

$$\xi_0 = \varepsilon^{2d+2} \;\; ; \quad \xi_i = (1+\varepsilon)\xi_{i-1} \;\; , \quad 1 \le i \le q \;\; ; \quad q := \left\lfloor \frac{\lg(\eta_f \eta_g \varepsilon^{-2(d+1)})}{\lg(1+\varepsilon)} \right\rfloor + 1 \; . \tag{36}$$

In view of the above, 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,

$$\hat{I} = \left\{ \hat{\mathbf{x}}^i = \Phi\mathcal{H}\left(\frac{\mathbf{x}^i}{\Phi}\right) : \; \mathbf{x}^i \in \tilde{I} \right\} \; , \tag{37}$$

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.

The proof of the following theorem is omitted due to space restrictions.

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

$$(1 - C_1\varepsilon)F(\hat{A}) \le F(\tilde{A}) \le (1 + C_1\varepsilon)F(\hat{A}) \ , \tag{38}$$

where C_1 is given in (8).

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

$$W = \mathcal{X}^d \quad \text{where} \quad \mathcal{X} = \{0, \xi_0, \ldots, \xi_{q-1}\} \ . \tag{39}$$

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

$$W = \{\mathbf{w}^1, \ldots, \mathbf{w}^s\} \ . \tag{40}$$

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

$$\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_s) \quad \text{where} \quad \mathbf{z}_i = \#\{\hat{\mathbf{x}} \in \hat{I} \ : \ \hat{\mathbf{x}} = \mathbf{w}^i\} \ , \ 1 \le i \le s \ . \tag{41}$$

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 such an assignment. 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

$$\emptyset = \hat{I}^0 \subseteq \hat{I}^1 \subseteq \ldots \subseteq \hat{I}^{m-1} \subseteq \hat{I}^m = \hat{I} \tag{42}$$

and

$$\mathbf{0} = \mathbf{z}^0 \le \mathbf{z}^1 \le \ldots \le \mathbf{z}^{m-1} \le \mathbf{z}^m = \mathbf{z} \ , \tag{43}$$

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 $Z = \{\mathbf{y} \ : \ \mathbf{0} \le \mathbf{y} \le \mathbf{z}\}$. With this, we define the graph $G = (V, E)$ as follows:

- The set of vertices consists of $m+1$ layers, $V = \cup_{k=0}^m V^k$. 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 $V^k = Z$, $0 < k < m$.
- The set of edges consists of m subsets:

  $$E = \cup_{k=1}^m E^k \quad \text{where} \quad E^k = \{(\mathbf{u}, \mathbf{v}) \ : \ \mathbf{u} \in V^{k-1} \ , \ \mathbf{v} \in V^k \ , \ \mathbf{u} \le \mathbf{v}\} \ . \tag{44}$$

  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:

$$t = |Z| = \prod_{i=1}^{s} (\mathbf{z}_i + 1) \le (n+1)^s \ . \tag{45}$$

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

$$w(e) = g^k(T(\mathbf{v} - \mathbf{u})) \quad \text{where} \quad T(\mathbf{v} - \mathbf{u}) = \sum_{i=1}^{s} (\mathbf{v}_i - \mathbf{u}_i)\mathbf{w}^i \ , \tag{46}$$

$\mathbf{w}^i$ are as in (40). We continue to define a cost function on the vertices, $r : V \to \mathcal{R}^+$. The cost function is defined recursively according to the layer of the vertex, 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 (6)). This cost function coincides with the cost function of the VAP, (2). 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 $e^k := (v^{k-1}, v^k) \in E^k,\ 1 \le k \le m$ 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 $|V| \le 2 + (m-1) \cdot (n+1)^s$ and $|E| = \sum_{k=1}^{m} |E^k| \le m \cdot (n+1)^{2s}$ 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 (7) and Lemma 1.
- Second modification: $\bar{I}$ to $\tilde{I}$, see (20)-(23) 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.

The proof of the following theorem is omitted due to space restrictions.

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 (1) with a constant that depends only on η_g, M_g and M_f.*

References

1. N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms*, pages 493–500, 1997.
2. A.K. Chandra and C.K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM Journal on Computing*, 4(3):249–263, 1975.
3. C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 185–194, 1999.
4. R.A. Cody and E.G. Coffman, Jr. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *J. Assoc. Comput. Mach.*, 23(1):103–115, 1976.
5. E. G. Coffman, Jr. and George S. Lueker. Approximation algorithms for extensible bin packing. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 586–588, 2001.
6. J. Csirik, H. Kellerer, and G. Woeginger. The exact lpt-bound for maximizing the minimum completion time. *Operations Research Letters*, 11:281–287, 1992.
7. W. F. de la Vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
8. 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*, 65(5):229–233, 1998.
9. P. Dell'Olmo and M. G. Speranza. Approximation algorithms for partitioning small items in unequal bins to minimize the total size. *Discrete Applied Mathematics*, 94:181–191, 1999.
10. M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory (Series A)*, 21:257–298, 1976.
11. R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
12. R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:416–429, 1969.
13. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.
14. N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one–dimensional bin–packing problem. In *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, 1982.
15. S. Sahni. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, 23:116–127, 1976.
16. G. J. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20:149–154, 1997.
17. G. J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters*, 64(6):293–297, 1997.