Journal of the Operations Research
Society of Japan
Vol. 32, No. 2, June 1989

PROPERTY OF THE OPTIMUM RELAXED SOLUTION FOR PROBLEM TO SCHEDULE INDEPENDENT TASKS ON UNRELATED PROCESSORS

Kazumiti Numata
The University of Electro-Communications

(Received May 27, 1988; Revised August 27, 1988)

Abstract The linear programming relaxation of the zero-one optimization problem to schedule n independent tasks nonpreemptively on m unrelated processors with the objective of minimizing the maximum completion time is considered. This relaxed problem is solved by mapping tasks into $m-1$ dimensional unit simplex $\mathcal{A}_{m-1}$ and by dividing it into m sub-simplexes (hyper cones), where each processor processes the tasks or the fractions of tasks whose images locate inside (including boundaries) of corresponded sub-simplex. The properties of the division of $\mathcal{A}_{m-1}$ (the associated partition of tasks) which generates the optimum relaxed solution and the existence of such a division are shown as two Theorems. They are proved, first, abstractly using Duality Theorem, and then directly. The latter elementary but intuitive proofs give helpful informations to find the optimum division of $\mathcal{A}_{m-1}$ directly. Finally applications of these theorems are discussed.

1. Introduction

We are given a set $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ of n independent tasks and a set $\mathcal{P} = \{P_1, P_2, \dots, P_m\}$ of m unrelated processors. Task $T_i (i = 1, 2, \dots, n)$ becomes available for execution at time zero and requires a positive processing time μ_{ij} if it is scheduled on processor $P_j (j = 1, 2, \dots, m)$. When processors are identical, then μ_{ij} is constant with respect to j i.e. is equal to r_i (processing requirement of task T_i). When processors are uniform, then μ_{ij} is expressed by using the notion of processor's speed (s_j) as r_i/s_j. In general case, matrix (μ_{ij}) has mn arbitrary positive entries. There is no precedence relations among tasks. A processor can work on only one task at a time, and a task can be worked on by any (one) processor. Tasks are processed nonpreemptively, i.e., once a task having started execution it will not be interrupted until its completion.

The problem is to schedule $\mathcal{T}$ on $\mathcal{P}$ so that the maximum completion time is minimized. Such a schedule is called an optimum schedule. The decision problem of determining whether $\mathcal{T}$ can be processed within a given finishing time is known to be $\boldsymbol{NP}$-complete even in the case of two($m = 2$) identical processors[1,3]. Hence it is unlikely that the polynomial time exact algorithm can be found to solve the problem. So the investigation has been directed to heuristic algorithms which provide an approximate solution.

Horowitz and Sahni[4] proposed an ϵ-approximate algorithm of time complexity $\mathcal{O}(n^{2m}/\epsilon)$ which can generate schedules arbitrarily close to the optimum for general m. Several heuristics were analyzed by Ibarra and Kim[5]. They also presented an $n\,log\,n$ time δ-approximate (with fixed worst-case performance ratio $1+\delta$) algorithm for $m = 2$ processors case, which is guaranteed to be at most $(1+\sqrt{5})/2$ times worse than the optimum ($\delta \cong 0.6$). Davis and Jaffe[2] presented several heuristics using the notion of efficiency for general m, and proved them to be at most $2\sqrt{m} \sim 1.5\sqrt{m}$ times worse than the optimum. Potts[10] developed a linear programming based heuristic for general m. This algorithm has a worst-case performance ratio of 2 for $m > 3$, and a modified version of it for $m = 2$ has a ratio 1.5. Lenstra et al.[7] presented an algorithm based on the similar idea to [10] which has better time complexity and the same error bound. Recently, Numata[9] proposed an ϵ-approximate algorithms for the case $m = 2$ which runs faster than [4] and generates more precise schedules than [5,7,10].

Most of these works, explicitly or implicitly, make use of the linear programming relaxation in order to determine an initial solution, to estimate errors, or to guide a heuristic procedure, where inherent properties of the relaxed optimum solution to this problem plays an essential role. Of course, it is necessary to compute numerical solutions. But such general methods as simplex method or alternative ones take not a little running time to find a solution, so fast algorithms based on the special structure of the problem (i.e. solution) are desired by approximate heuristics which solve relaxed problems repeatedly many times. From these points of view, few researches are known except for the case of $m = 2$.

In this paper we consider the linear programming relaxation problem of the general unrelated processors system. In section 2 by normalizing m-tuple $(\mu_{i1}, \mu_{i2}, \ldots, \mu_{im})$, task T_i is mapped into the $m-1$ dimensional unit simplex $\mathcal{A}_{m-1}$ in $\boldsymbol{R}^m$ $(i = 1, 2, \ldots, n)$. In section 3 we divide $\mathcal{A}_{m-1}$ into m sub-simplexes, which correspond to m processors, under a certain constraint, where the tasks or the task-fractions whose images locate inside of a sub-simplex are assigned to the corresponded processor, and those on a boundary are shared at certain ratio by the processors (sub-simplexes) which constitute the boundary. Then, we present Theorem 1 that asserts the existence of the division which satisfies specified conditions and Theorem 2 that assures such a division generates the optimum relaxed solution. In section 4, first, these theorems are proved abstractly using Duality Theorem, and then directly. The latter elementary but intuitive proofs are helpful to guide the algorithm which directly determines the (near) optimum division of $\mathcal{A}_{m-1}$. Applications of these theorems are discussed in section 5.

2. Preliminaries

When $m = 2$, using continuous variable x_i in the interval [0,1], which represents that x_i portion of T_i is processed by P_1 and that $(1 - x_i)$ portion by P_2 ($i = 1, 2, \ldots n$), our relaxed problem is written as follows.

$$(2.1)\quad \begin{cases} \text{minimize} & y \\ \text{subject to} & \sum_{i=1}^{n} \mu_{i1} \cdot x_i \le y \\ & \sum_{i=1}^{n} \mu_{i2} \cdot (1 - x_i) \le y \\ & y \ge 0, \quad 0 \le x_i \le 1 \quad (i = 1, 2, \ldots, n). \end{cases}$$

In [6], execution times list (μ_{i1}, μ_{i2}) of T_i is regarded as coordinates of a point and is plotted on a plane $\boldsymbol{R}^2$. Then, solution of (2.1) is found by using the slopes of lines connecting the origin and these points as the index (Fig.1). Instead of the slope, we can use the point $(\mu_{i1}/(\mu_{i1} + \mu_{i2}), \mu_{i2}/(\mu_{i1} + \mu_{i2}))$ as an equivalent index. This corresponds to projecting tasks to points on the segment connecting (0,1) and (1,0) (Fig.2).

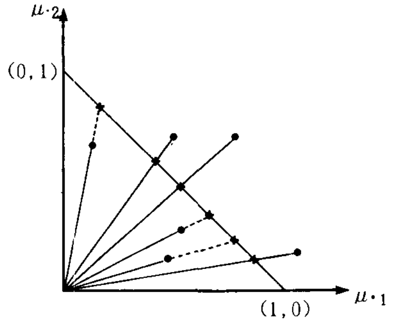


Fig.1 point representation of tasks

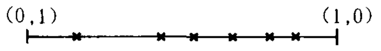


Fig.2 normalized tasks

The solution is to assign points (tasks) to P_1 from the side of (0,1) and to P_2 from (1,0) one by one so that the processing times of both processors may become equal, where the last assigned task may be divided to both P_1 and P_2 at some ratio (see Fig.3).

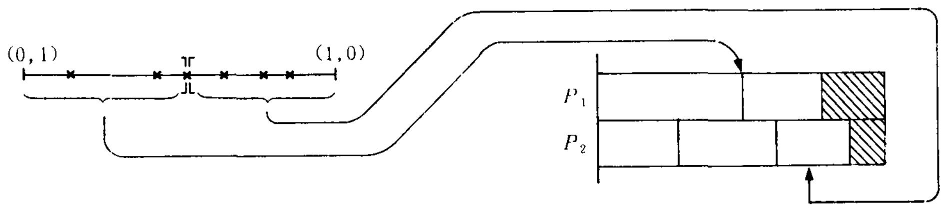


Fig.3 relaxed solution of the case $m = 2$

When increasing the number of processors to $m = 3$, we can consider the problem in the same way by mapping tasks to points on the equilateral triangle (2-dimensional simplex $\mathcal{A}_2$) that consists of three vertices (0,0,1) (0,1,0) and (1,0,0) (Fig.4).

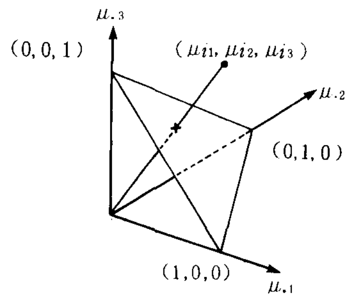


Fig.4 case of $m = 3$

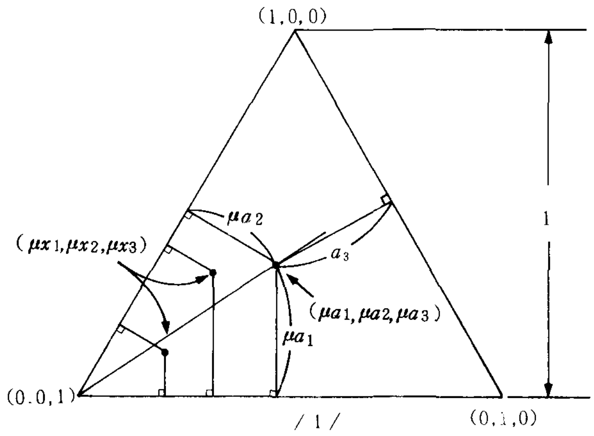


Fig.5 representation in $\mathcal{A}_2$

The mapping (projection) φ from $\mathcal{T}$ to $\mathcal{A}_2$ is defined as:

$$\varphi : \; T_i = (\mu_{i1}, \mu_{i2}, \mu_{i3}) \in \mathcal{T} \quad \longmapsto \quad \left(\frac{\mu_{i1}}{s_i}, \frac{\mu_{i2}}{s_i}, \frac{\mu_{i3}}{s_i}\right), \quad \text{where} \quad s_i = \textstyle\sum_{j=1}^{3} \mu_{ij}\,.$$

This expression of a task on $\mathcal{A}_2$ generally is called "area coordinates" or "barycentric coordinates". When the image of task T_a is given by

$$\varphi(T_a) = (\mu_{a1}, \mu_{a2}, \mu_{a3}) \quad (\text{where } \textstyle\sum_{j=1}^{3} \mu_{aj} = 1),$$

then, the point which is located at distances (heights) μ_{a1}, μ_{a2} and μ_{a3} from three edges of $\mathcal{A}_2$ (the height of the equilateral triangle is assumed to be 1),

$$\begin{array}{ll} /1/ : \overline{(0,0,1)\text{—}(0,1,0)} & \text{(1st coordinate is fixed to 0),} \\ /2/ : \overline{(1,0,0)\text{—}(0,0,1)} & \text{(2nd coordinate is fixed to 0) and} \\ /3/ : \overline{(0,1,0)\text{—}(1,0,0)} & \text{(3rd coordinate is fixed to 0)} \end{array}$$

corresponds to T_a. Each edge of $\mathcal{A}_2$ is isomorphic to $\mathcal{A}_1$ (Fig.5). Task T_x whose image, $\varphi(T_x) = (\mu_{x1}, \mu_{x2}, \mu_{x3})$, satisfies the relation

$$\mu_{xj}/\mu_{xk} = \mu_{aj}/\mu_{ak} \quad j,k \in \{1,2,3\},\ j \neq k$$

is on the straight line which passes a point of the intersection of $/j/$ and $/k/$, and has the ratio $\mu_{aj} : \mu_{ak}$ of distances to $/j/$ and $/k/$ (distances mean the lengths of perpendiculars from a point on the line to $/j/$ and $/k/$). If $\mu_{xj}/\mu_{xk} < \mu_{aj}/\mu_{ak}$ then T_x lies in $/j/$'s side of this line, and if $\mu_{xj}/\mu_{xk} > \mu_{aj}/\mu_{ak}$ then in $/k/$'s side.

When the dimension (number of processors) is increased as $m = 4, 5, \ldots$ the above idea is naturally extended. In the case of $m = 4$, $\mathcal{A}_3$ is given by a regular tetrahedron (Fig.6), and each task is expressed by "volume coordinates".

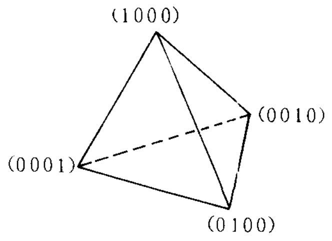


Fig.6 case of $m = 4$

For general m, $\mathcal{A}_{m-1}$ is an $m-1$ dimensional unit simplex in R^m and determined as

$$\mathcal{A}_{m-1} = \{ (\xi_1, \xi_2, \dots, \xi_m) \mid \sum_{j=1}^{m} \xi_j = 1,\ \xi_j \geq 0 \}.$$

The mapping from $\mathcal{T}$ to $\mathcal{A}_{m-1}$ is given by

$$\varphi :\ T_i = (\mu_{i1}, \mu_{i2}, \dots, \mu_{im}) \in \mathcal{T} \quad \longmapsto \quad (\frac{\mu_{i1}}{s_i}, \frac{\mu_{i2}}{s_i}, \dots, \frac{\mu_{im}}{s_i}), \quad \text{where } s_i = \textstyle\sum_{j=1}^{m} \mu_{ij}.$$

Here, we consider $(m-2)$-dimensional faces (edge simplexes) of $\mathcal{A}_{m-1}$ [11]. Each of them is isomorphic to $\mathcal{A}_{m-2}$ and consists of remaining $m-1$ vertices after exclusion of some one vertex out of m vertices of $\mathcal{A}_{m-1}$. The face obtained by deleting the vertex $(0, \dots, 0, 1^{(j)}, 0, \dots, 0)$ is designated by $/j/$ and expressed as follows.

$$/j/ = \{ (\xi_1, \dots, \xi_{j-1}, 0, \xi_{j+1}, \dots, \xi_m) \mid \sum_{k=1}^{m} \xi_k = 1,\ \xi_k \geq 0 \}$$

When the image of task T_a is given by $\varphi(T_a) = (\mu_{a1}, \mu_{a2}, \dots, \mu_{am})$, in the same way as the case $m = 3$, it corresponds to the point located at the distance (height) μ_{aj} from face $/j/$ $(j = 1, 2, \dots, m)$, where the height of $\mathcal{A}_{m-1}$ (the distance between an face and the excluded vertex) is assumed to be 1. Task T_x whose image, $\varphi(T_x) = (\mu_{x1}, \mu_{x2}, \dots, \mu_{xm})$, satisfies the relation

$$\mu_{xj}/\mu_{xk} = \mu_{aj}/\mu_{ak} \quad j, k \in \{1, 2, \dots, m\},\ \ j \neq k$$

is on the hyperplane which passes points of the intersection of $/j/$ and $/k/$, and has the ratio $\mu_{aj} : \mu_{ak}$ of distances from it to $/j/$ and $/k/$. If $\mu_{xj}/\mu_{xk} < \mu_{aj}/\mu_{ak}$ then T_x lies in $/j/$'s side of this hyperplane, and if $\mu_{xj}/\mu_{xk} > \mu_{aj}/\mu_{ak}$ then in $/k/$'s side.

3. Definitions and Theorems

We call a part of a task "task-fraction", and let $\theta \cdot T_i$ denote the θ part of T_i $(0 \leq \theta \leq 1)$.

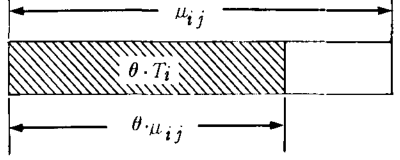


Fig.7 task-fraction

A part of a task-fraction is denoted by $\theta_2 \cdot (\theta_1 \cdot T_i) = (\theta_1\theta_2) \cdot T_i$ and again is called task-fraction. Generally we consider that $\theta_k \cdot (\theta_{k-1} \cdot (\cdots (\theta_1 \cdot T_i) \cdots)) = \theta_k\theta_{k-1}\cdots\theta_1 \cdot T_i$, where $1 \cdot T_i$ means T_i itself, and $0 \cdot T_i$ a null task. Here after, let $\mu_j(\cdot)$ denote the time requirement for processor P_j to process a tasks, a task-fraction or a set of them, $U = \left\{ \theta_i \cdot T_i \,|\, i \in I \subset \{1,2,\ldots,n\} \right\}$. For example, we have

$$\begin{aligned} \mu_j(T_i) &= \mu_{ij} \\ \mu_j(\theta \cdot T_i) &= \theta \cdot \mu_j(T_i) = \theta\mu_{ij} \\ \mu_j(U) &= \sum_{i\in I} \theta_i \cdot \mu_j(T_i) = \sum_{i\in I} \theta_i\mu_{ij}\,. \end{aligned}$$

For a set of task-fractions U, we define its similar reduction $U(\theta)$ by

$$U(\theta) = \{\, \theta \cdot \theta_i \cdot T_i \,|\, i \in I \,\}\,.$$

$U(\theta)$ is a set of θ parts of all task-fractions of U, and has the same composition ratios in tasks as U. On any processor P_j ($j \in \{1,2,\ldots,m\}$), we have

$$\mu_j(U(\theta)) = \theta \cdot \mu_j(U)\,.$$

Next, let a feasible solution $\mathcal{X} = (x_{ij})$ of the relaxed problem

$$(3.1)\qquad \begin{cases} \text{minimize} & y \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} \mu_{ij} \cdot x_{ij} \le y \qquad j = 1,2,\ldots,m \\ & \displaystyle\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1,2,\ldots,n \\ & x_{ij} \ge 0 \qquad i = 1,2,\ldots,n\,; \quad j = 1,2,\ldots,m \end{cases}$$

correspond to the partition of $\mathcal{T}$, ($\{V_j\}$ ($j = 1,2,\ldots,m$) by

$$V_j = \{x_{ij} \cdot T_i \,|\, i = 1,2,\ldots,n\} \qquad j = 1,2,\ldots,m\,.$$

Since x_{ij} denote the ratio of T_i which is processed on P_j, so V_j means the set of all task-fractions that is processed by P_j.

The schedule length of the partition $\mathcal{X} \cong \{V_j\}$ is noted and defined as length$(\mathcal{X}) = \max_j \mu_j(V_j)$. A partition is said to be "even", if

$$\mu_1(V_1) = \mu_2(V_2) = \cdots = \mu_m(V_m)\,.$$

Now we consider the hyper-cone (sub-simplex) $\mathcal{C}_j(\vec{\alpha})$ whose vertex is given by the point $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ in $\mathcal{A}_{m-1}$ and which has the face $/j/$ as its base ($j = 1,2,\ldots,m$). In other words, $\mathcal{C}_j(\vec{\alpha})$ is a intersection for $k = 1,2,\ldots,m (k \neq j)$ of the $/j/$'s side of two parts of $\mathcal{A}_{m-1}$ divided by the hyper plane which includes the intersection of faces $/j/$ and $/k/$ and has the ratio $\alpha_j : \alpha_k$ of distances to $/j/$ and $/k/$. So it is written as

$$\mathcal{C}_j(\vec{\alpha}) = \left\{ (\xi_1, \xi_2, \ldots, \xi_m) \;\middle|\; \sum_{k=1}^{m} \xi_k = 1 \;\text{ and }\; \bigwedge_{k\neq j} \left(\frac{\xi_j}{\xi_k} \le \frac{\alpha_j}{\alpha_k}\right) \right\}.$$

Then, all tasks of $\mathcal{T}$ distributed in $\mathcal{A}_{m-1}$ is partitioned by these hyper-cones, that is, V_j is given by a set of tasks (task-fractions) whose images belong to $C_j(\vec{\alpha})$. Because the base of $C_j(\vec{\alpha})$ is

$$/j/ = \{(\xi_1,\cdots,\xi_{j-1},0,\xi_{j+1},\cdots,\xi_m) \mid \sum_{k=1}^{m}\xi_k = 1,\ \xi_k \geq 0\},$$

it is natural to assign tasks near this face to processor P_j. Such a partition as this is called $\vec{\alpha}$-simple one. More strictly, a task partition $\{V_j\}$ of the given $\mathcal{T}$ is $\vec{\alpha}$-simple, if and only if

$$\begin{cases} \varphi(T_i) \in \mathrm{Int}(C_j(\vec{\alpha})) \implies T_i \in V_j \quad \text{i.e. } x_{ij} = 1 \\ \varphi(T_i) \in C_{j_1}(\vec{\alpha}) \cap C_{j_2}(\vec{\alpha}) \cap \cdots \cap C_{j_e}(\vec{\alpha}) \implies \\ \qquad \theta_{i,j_s} \cdot T_i \in V_{j_s} \quad \text{i.e. } x_{i,j_s} = \theta_{i,j_s}, \quad \text{where} \quad \sum_{s=1}^{e} \theta_{i,j_s} = 1 \,. \end{cases}$$

Fig.8 shows two partitions of the case $m = 3$, where (a) is $\vec{\alpha}$-simple but (b) is not.

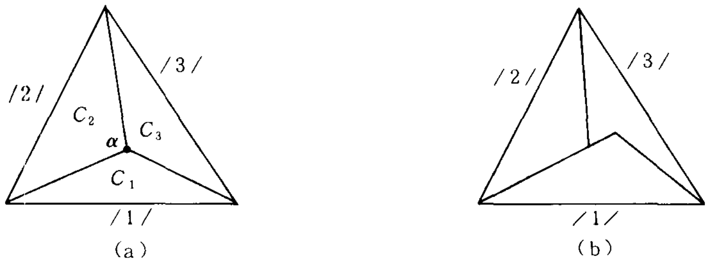


Fig.8 $\vec{\alpha}$-simple partitions

Since a task on a boundary may belong at arbitrary ratios to more than one V_js corresponded to sub-simplexes (processors) which share the boundary, it should be noted that infinitely many $\vec{\alpha}$-simple partitions exist for a fixed $\vec{\alpha}$.

After the above preparations, we have the following theorems.

Theorem 1. *For any $\mathcal{T}$ and $\mathcal{P}$, there exists such a point $(\alpha_1,\alpha_2,\cdots,\alpha_m) \in \mathcal{A}_{m-1}$ that realizes the $\vec{\alpha}$-simple and even partition $\mathcal{X} = (x_{ij}) \cong \{V_j\}$ of $\mathcal{T}$.*

Theorem 2. *The $\vec{\alpha}$-simple and even partition $\mathcal{X} = (x_{ij}) \cong \{V_j\}$ is the optimum solution of the relaxed problem* (3.1).

Theorem 2 describes the sufficient condition of the optimum relaxed solution in terms of the task partition in $\mathcal{A}_{m-1}$, and Theorem 1 asserts the existence of the partition that satisfies this condition.

4. Proofs

In this section we prove Theorem 1 and 2, assuming without loss of generality that

$$\forall T_i \in \mathcal{T}\,,\ \varphi(T_i) \in \operatorname{Int}(\mathcal{A}_{m-1})\,, \quad \text{i.e.} \quad \forall i,j \quad \mu_{ij} > 0. \tag{4.1}$$

If $\mu_{ij} = 0$ then it suffices to assign T_i to P_j (this assignment has no effect on processors) and consider the remaining $\mathcal{T} - \{T_i\}$.

4.1 Proof by Duality Theorem

Problem (3.1) is rewritten as

$$\tag{4.2} \begin{cases} \text{minimize} & z = y \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} -\mu_{ij}\cdot x_{ij} + y \geq 0 \qquad j = 1,2,\ldots,m \\ & \displaystyle\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1,2,\ldots,n \\ & x_{ij} \geq 0 \qquad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m\,. \end{cases}$$

Associating dual variables $\xi_1, \xi_2, \ldots, \xi_m$ with first m constraints of (4.2) and $\eta_1, \eta_2, \ldots, \eta_n$ with remaining n (equality) constraints, we have the following dual problem

$$\tag{4.3} \begin{cases} \text{maximize} & \omega = \eta_1 + \eta_2 + \cdots + \eta_n \\ \text{subject to} & -\mu_{ij}\cdot\xi_j + \eta_i \leq 0 \qquad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m \\ & \displaystyle\sum_{j=1}^{m} \xi_j = 1 \\ & \xi_j \geq 0 \qquad j = 1,2,\ldots,m\,. \end{cases}$$

Since (4.2) has the bounded optimum solution, so (4.3) also does. Let $\hat{z}$, $\check{y}$ and $\hat{x}_{ij}$ $(i = 1,2,\ldots,n;\, j = 1,2,\ldots,m)$ be the optimum solution of (4.2) and $\check{\omega}$, $\check{\xi}_j$ $(j = 1,2,\ldots,m)$ and $\check{\eta}_i$ $(i = 1,2,\ldots,n)$ be that of (4.3), where $\hat{z} = \check{\omega}$. And let $\hat{V}_j$ be the set of task-fractions $\{\, \hat{x}_{ij}\cdot T_i \,|\, i = 1,2,\ldots,n \,\}$.

First mn constraints of (4.3) are equivalent to

$$\eta_i \leq \min(\mu_{i1}\xi_1, \mu_{i2}\xi_2, \cdots, \mu_{im}\xi_m) \qquad i = 1,2,\ldots,n.$$

Since the objective function of (4.3), $\omega = \sum_{i=1}^{n} \eta_i$ is maximized, these constraints must be satisfied with equality at the optimum point. That is, it holds that

$$\check{\eta}_i = \min(\,\mu_{i1}\check{\xi}_1, \mu_{i2}\check{\xi}_2, \cdots, \mu_{im}\check{\xi}_m\,) \qquad i = 1,2,\ldots,n\,. \tag{4.4}$$

Furthermore, we have

$$\forall j \qquad \check{\xi}_j > 0\,. \tag{4.5}$$

Because, if $\check{\xi}_j = 0$ for some j then we have $\check{\eta}_i = 0$ for all i by (4.4), hence $\check{\omega} = 0$. But (4.1) implies that $\hat{z} > 0$. These contradict $\check{\omega} = \hat{z}$.

Here, Duality Theorem (complementary slackness condition, see e.g. [6]) asserts that

$$\check{\xi}_j \cdot \Bigl(-\sum_{i=1}^{n} \mu_{ij}\,\hat{x}_{ij} + \hat{y}\Bigr) = 0 \qquad j = 1,2,\ldots,m\,, \tag{4.6}$$

and that

$$\hat{x}_{ij} \cdot (-\mu_{ij}\,\check{\xi}_j + \check{\eta}_i) = 0 \qquad i = 1,2,\ldots,n\,;\; j = 1,2,\ldots,m\,. \tag{4.7}$$

From (4.5) and (4.6) it holds that

$$\hat{y} = \sum_{i=1}^{n} \mu_{ij}\,\hat{x}_{ij} \qquad j = 1,2,\ldots,m\,,$$

that is,

$$\mu_1(\hat{V}_1) = \mu_2(\hat{V}_2) = \cdots = \mu_m(\hat{V}_m)\,. \tag{4.8}$$

Next, let $\hat{\vec{\alpha}}$ be $\varphi(1/\check{\xi}_1, 1/\check{\xi}_2, \ldots, 1/\check{\xi}_m)$ (it is possible by (4.5)), that is,

$$\hat{\vec{\alpha}} = \Bigl(\frac{A}{\check{\xi}_1}, \frac{A}{\check{\xi}_2}, \ldots, \frac{A}{\check{\xi}_m}\Bigr) \qquad \text{where } A = 1\Big/\textstyle\sum_{j=1}^{m} \frac{1}{\check{\xi}_j}\,.$$

The hyper-cone (sub-simplex) $\mathcal{C}_h(\hat{\vec{\alpha}})$ determined by vertex $\hat{\vec{\alpha}}$ and base $/h/$ is written as: (by reversing the inequality symbol and ratios of both sides of the definition formula of $\mathcal{C}_j(\vec{\alpha})$ in section 3)

$$\mathcal{C}_h(\hat{\vec{\alpha}}) = \Bigl\{ (\zeta_1, \zeta_2, \ldots, \zeta_m) \;\Big|\; \sum_{j=1}^{m} \zeta_j = 1 \quad \text{and} \quad \bigwedge_{j \neq h} \Bigl(\frac{\zeta_j}{\zeta_h} \geq \frac{\check{\xi}_h}{\check{\xi}_j}\Bigr) \Bigr\}\,.$$

If $\varphi(T_i)$ belongs to the interior of $\mathcal{C}_h(\hat{\vec{\alpha}})$, we have

$$\begin{aligned} &\forall j\,(\neq h) \quad \frac{\mu_{ij}}{\mu_{ih}} > \frac{\check{\xi}_h}{\check{\xi}_j} \\ \Longrightarrow\; &\forall j(\neq h) \quad \mu_{ij}\check{\xi}_j > \mu_{ih}\check{\xi}_h \\ \overset{(4.4)}{\Longrightarrow}\; &\check{\eta}_i = \mu_{ih}\check{\xi}_h \quad \text{and} \quad \forall j(\neq h) \quad \check{\eta}_i < \mu_{ij}\check{\xi}_j \\ \overset{(4.7)}{\Longrightarrow}\; &\forall j(\neq h) \quad \hat{x}_{ij} = 0\,. \end{aligned} \tag{4.9}$$

Generally it holds that

$$\begin{aligned} &\varphi(T_i) \in \mathcal{C}_{h_1}(\hat{\vec{\alpha}}) \cap \mathcal{C}_{h_2}(\hat{\vec{\alpha}}) \cap \cdots \cap \mathcal{C}_{h_e}(\hat{\vec{\alpha}}) \\ \Longrightarrow\; &\forall s,t \in \{1,2,\ldots,e\} \quad \frac{\mu_{i,h_s}}{\mu_{i,h_t}} = \frac{\check{\xi}_{h_t}}{\check{\xi}_{h_s}} \quad \text{and} \\ &\forall j \notin \{h_1, h_2, \ldots, h_e\}\,, \quad \forall s \in \{1,2,\ldots,e\} \qquad \frac{\mu_{ij}}{\mu_{i,h_s}} > \frac{\check{\xi}_{h_s}}{\check{\xi}_j} \\ \Longrightarrow\; &\check{\eta}_i = \mu_{i,h_1}\check{\xi}_{h_1} = \mu_{i,h_2}\check{\xi}_{h_2} = \cdots = \mu_{i,h_e}\check{\xi}_{h_e} \quad \text{and} \\ &\forall j \notin \{h_1, h_2, \ldots, h_e\} \qquad \check{\eta}_i < \mu_{ij}\check{\xi}_j \\ \Longrightarrow\; &\forall j \notin \{h_1, h_2, \ldots, h_e\} \qquad \hat{x}_{ij} = 0 \quad \text{i.e.} \quad \sum_{s=1}^{e} \hat{x}_{i,h_s} = 1\,. \end{aligned} \tag{4.10}$$

The existence of $\hat{\vec{\alpha}} = (A/\check{\xi}_1, A/\check{\xi}_2, \ldots, A/\check{\xi}_m)$ and (4.8),(4.9) and (4.10) complete the proofs of Theorem 1 and 2 at the same time. □

4.2 Another proof of Theorem 1

Instead of proving Theorem 1 directly, we consider the following Lemma 1, which introduces a parameter l related to the number of fixed boundaries between two sub-simplexes in $\mathcal{A}_{m-1}$. When $l = 1$, it coincides with Theorem 1. Lemma 1 is proved through a mathematical (backward) induction on l.

Lemma 1. *For any $\mathcal{T}$ and $\mathcal{P}$, there exists such a point*

$$\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{l-1}, \alpha_l, \ldots, \alpha_m) \in \mathcal{A}_{m-1} \qquad (\text{where } \alpha_1 \cdots \alpha_m \neq 0)$$

that can realize the partition of $\mathcal{T}$ which is $\vec{\alpha}$-simple and even as for $l \sim m$, i.e. $\mu_l(V_l) = \mu_{l+1}(V_{l+1}) = \cdots = \mu_m(V_m)$, under the condition that mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_l$ $(\neq 0)$ are arbitrarily fixed and the percentages with which each of tasks on boundaries among $\mathcal{C}_1 \sim \mathcal{C}_l$ is shared by corresponded V_js (P_js) are also arbitrarily fixed.

proof of Lemma 1 : First, we see that this Lemma holds when $l = m-1$. In this case, only $\delta = \sum_{j=1}^{m-1} \alpha_j$ (or $1 - \delta = \alpha_m$, $0 < \delta < 1$), i.e. boundaries between $\mathcal{C}_m(\vec{\alpha})$ and $\mathcal{C}_j(\vec{\alpha})$ $(j = 1, 2, \ldots, m-1)$ are movable. It is shown that $\mu_{m-1}(V_{m-1})$ and $\mu_m(V_m)$ can be equalized by adjusting the value of δ. Hereafter any task on mutual boundaries among $\mathcal{C}_1 \sim \mathcal{C}_{m-1}$ is assumed to be divided into sub-tasks according to the percentages with which it is shared by corresponded V_js (P_js).

Let δ become sufficiently small. More precisely, let it take the value δ^0 which satisfies

$$0 < \delta^0 < \frac{\omega}{1+\omega}, \qquad \text{where } \omega = \min_{ij} \frac{\mu_{ij}}{\mu_{im}}.$$

Assumption (4.1) guarantees the existence of such a value. For this δ^0,

$$\forall i\,(1 \le i \le n),\ \forall j\,(1 \le j \le m-1) \qquad \frac{\alpha_j}{\alpha_m} < \frac{\delta^0}{1-\delta^0} = \omega \le \frac{\mu_{ij}}{\mu_{im}}$$

holds, then all tasks belong to $\mathcal{C}_m(\vec{\alpha})$. Therefore, we have

$$\begin{cases} \mu_m(V_m) > 0 \\ \mu_{m-1}(V_{m-1}) = 0\,. \end{cases}$$

Next, let δ increase from δ^0 little by little. During this operation, only the boundaries between $\mathcal{C}_m(\vec{\alpha})$ and $\mathcal{C}_j(\vec{\alpha})$ $(j = 1, 2, \ldots, m-1)$ are moved. For $\vec{\alpha}' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_m)$ determined by δ' and $\vec{\alpha}'' = (\alpha''_1, \alpha''_2, \ldots, \alpha''_m)$ determined by δ'', where $\delta' < \delta''$, we have

$$\forall j \in \{1, 2, \ldots, m-1\} \qquad \frac{\alpha'_j}{\alpha'_m} < \frac{\alpha''_j}{\alpha''_m}\,.$$

Consequently, all movable boundaries are moved toward $/m/$ when δ increases (Fig.9). In short, the region of $\mathcal{C}_m(\vec{\alpha})$ shrinks monotonously (Fig. 10). When $\delta'' - \delta' \to 0$, obviously $\alpha''_j/\alpha''_m - \alpha'_j/\alpha'_m \to 0$. So, $\mathcal{C}_m(\vec{\alpha})$ shrinks continuously on δ.

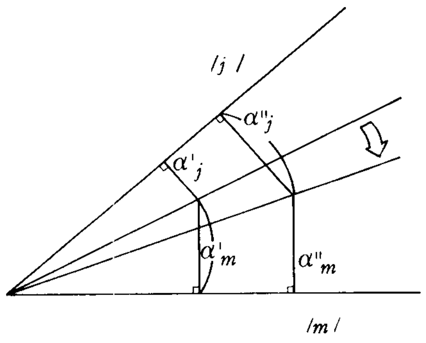


Fig.9 movement of a boundary

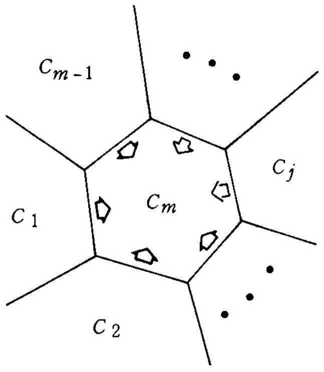


Fig.10 shrinkage of C_m

As δ is increased like this, it occurs that boundaries pass tasks. When a boundary meets a task, keeping δ stationary, we decrease the percentage with which the task belongs to V_m little by little. The task which at first entirely belongs to V_m is gradually shifted to the participant V_j of the boundary. If boundaries meet more than one tasks at the same time, these tasks are treated in sequence. When δ takes the value δ^1 which is sufficiently close to 1 at last, $C_m(\vec{\alpha})$ does not contain any task. So we have

$$\begin{cases} \mu_m(V_m) = 0 \\ \mu_{m-1}(V_{m-1}) \geq 0 . \end{cases}$$

While δ increases from δ^0 to δ^1 in the above manner, $\mu_m(V_m)$ and $\mu_{m-1}(V_{m-1})$ change continuously as Fig.11. $\mu_m(V_m)$ is a nonincreasing n-steps function of δ and $\mu_{m-1}(V_{m-1})$ is a nondecreasing n'-steps ($n' \leq n$) function of δ. Steps appear at the values of δ s where boundaries meet tasks. $\mu_m(V_m)$ or $\mu_{m-1}(V_{m-1})$ is multivalent for such δ s.

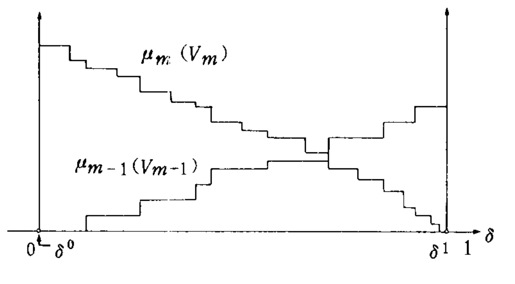


Fig.11 changes of $\mu_m(V_m)$ and $\mu_{m-1}(V_{m-1})$

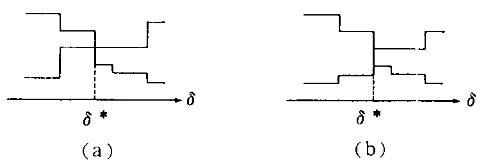


Fig.12 two types of the crossing

The graphs of $\mu_m(V_m)$ and $\mu_{m-1}(V_{m-1})$ intersects at $\delta^*(\delta^0 < \delta^* < \delta^1)$ in such a manner as either (a) or (b) of Fig.12. In case (a), $\vec{\alpha}$-simple partition uniquely determined by δ^* realizes $\mu_{m-1}(V_{m-1}) = \mu_m(V_m)$. In case (b), T_x is being shfted from V_m to V_{m-1} at $\delta = \delta^*$. Let ϵ $(0 \leq \epsilon \leq 1)$ be the percentage with which T_x belongs to V_{m-1}, then this process is shown as Fig.13.

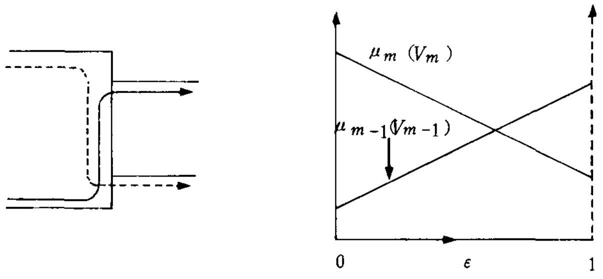


Fig.13 shifting of a task from V_m to V_{m-1} at $\delta = \delta^*$

From Fig.13 it can be seen that δ^* gives $\vec{\alpha}$-simple partition which realizes $\mu_{m-1}(V_{m-1}) = \mu_m(V_m)$ also in case (b).

Next, assuming that Lemma 1 holds for $l = m-1, m-2, \ldots, h$ (hypothesis of induction), we consider the case $l = h-1$. That is, we will show that there exists such a point of $\mathcal{A}_{m-1}$ that can realize the partition of $\mathcal{T}$ which is $\vec{\alpha}$-simple and even as for $h-1 \sim m$ i.e.

$$\mu_{h-1}(V_{h-1}) = \mu_h(V_h) = \mu_{h+1}(V_{h+1}) = \cdots = \mu_m(V_m),$$

under the condition that mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_{h-1}$ ($\neq 0$) are arbitrarily fixed and the percentages with which each of tasks on boundaries among $C_1 \sim C_{h-1}$ is shared by corresponded $V_j s$ ($P_j s$) are also arbitrarily fixed. Here we can only adjust $\sum_{j=1}^{h-1} \alpha_j$ (or $\sum_{j=h}^{m} \alpha_j$) and mutual ratios among $\alpha_h, \alpha_{h+1}, \ldots, \alpha_m$ ($\neq 0$). Any task on mutual boundaries among $C_1 \sim C_{h-1}$ is assumed to be divided into sub-tasks in the same way as $l = m-1$.

At first let $\delta = \sum_{j=1}^{h-1} \alpha_j$ be sufficiently small, and α_h be sufficiently large (relative to $\alpha_1 \sim \alpha_{h-1}$). More precisely, using the notation $\sigma = \sum_{j=1}^{h} \alpha_j$, we select the values of δ and α_h so that $\gamma = \delta/\sigma$ $(1-\gamma = \alpha_h/\sigma)$, satisfys

$$0 < \gamma < \frac{\omega}{1+\omega}, \qquad \text{where } \omega = \min_{1\le i\le n} \min_{1\le j\le h-1} \frac{\mu_{ij}}{\mu_{ih}}.$$

(This is possible from assumption (4.1).) For $\vec{\alpha}$ that has mutual ratios among $\alpha_1, \alpha_2, \ldots$, α_{h-1}, α_h determined as the above, no task belongs to any of $C_1(\vec{\alpha}), C_2(\vec{\alpha}), \cdots, C_{h-1}(\vec{\alpha})$. And, by hypothesis of induction, under these given ratios among $\alpha_1 \sim \alpha_h$, adjustment of mutual ratios among $\alpha_{h+1}, \alpha_{h+2}, \ldots, \alpha_m$ ($\neq 0$) realizes (see Fig.14)

$$\mu_h(V_h) = \mu_{h+1}(V_{h+1}) = \cdots = \mu_m(V_m). \tag{4.11}$$

We assume that (4.11) holds at $\vec{\alpha} = \vec{\alpha}^0$, and let δ^0 denote $\sum_{j=1}^{h-1} \alpha_j^0$. Here let $L_{h\sim m}$ be the value of $\mu_h(\cdot) \sim \mu_m(\cdot)$ when (4.11) is realized, that is

$$L_{h\sim m} = \mu_h(V_h) = \mu_{h+1}(V_{h+1}) = \cdots = \mu_m(V_m).$$

Now we have

$$\begin{cases} L_{h\sim m} > 0 \\ \mu_{h-1}(V_{h-1}) = 0\,. \end{cases}$$

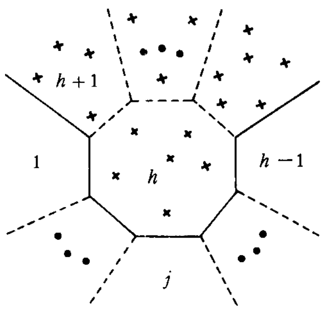


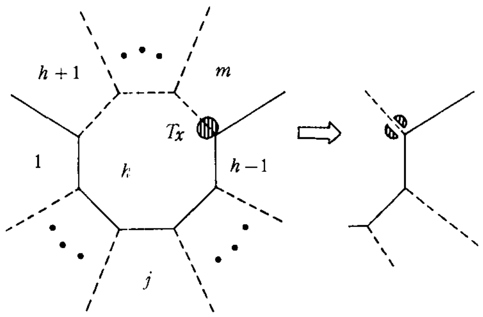
Fig.14 initial state

The following Operation 1 is applied to this partition state $(\vec{\alpha}^0, \delta^0, \mathcal{X} \cong \{V_j\})$.

Operation 1 *Increase the value $\delta = \sum_{j=1}^{h-1} \alpha_j$ (from δ^0) little by little, with keeping mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_{h-1}$ and those among $\alpha_h, \alpha_{h+1}, \ldots, \alpha_m$ constant.*

With this operation the boundaries between the group of $C_1 \sim C_{h-1}$ and the group of $C_h \sim C_m$ (represented by the bold line in Fig.14) are moved toward the sides of $/h/ \sim /m/$ little by little. Let us suppose that these boundaries first meet a task at $\delta = \delta^{00}$ and $\vec{\alpha} = \vec{\alpha}^{00}$. While the partition state is changed from $(\vec{\alpha}^0, \delta^0, \mathcal{X})$ to $(\vec{\alpha}^{00}, \delta^{00}, \mathcal{X})$ by Operation 1, no task crosses any boundary. So relation (4.11) is conserved.

Let T_x be the task which one of moving boundaries meets at $(\vec{\alpha}^{00}, \delta^{00}, \mathcal{X})$. If boundaries meet more than one tasks at the same time, these tasks are treated in sequence. We assume without loss of generality that T_x is on the boundary between $C_h(\vec{\alpha}^{00})$ and $C_a(\vec{\alpha}^{00})$ $(1 \le a \le h-1)$ (if otherwise then reindex $h \sim m$). If T_x is also on the boundary between some of $C_h \sim C_m$ and belongs to more than one of $V_h \sim V_m$, then T_x is assumed to be divided into sub-tasks according to the percentages with which it is shared by them (Fig.15).



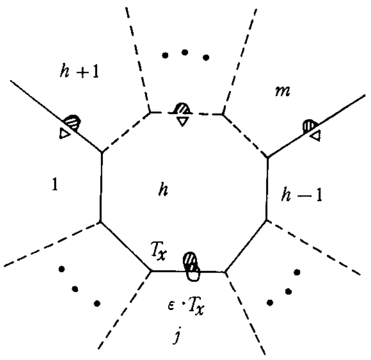


Fig.15 preparation Fig.16 shifting of tasks

After these preparations, we apply the following Operation 2 to the partition state $(\vec{\alpha}^{00}, \delta^{00}, \mathcal{X} \cong \{V_j\})$.

Operation 2 *If the partition state $(\vec{\alpha}, \delta, \mathcal{X} \cong \{V_j\}, L_{h\sim m})$ has other tasks than T_x which locate on the boundaries between C_a and C_b $(1 \le a \le h-1,\ h+1 \le b \le m)$, then* *their fractions belonging to V_a are moved a little toward $/a/$ off the boundary. In the same way, tasks on the boundaries between C_h and C_b (if any) are moved a little toward $/h/$ and the resultant decrease of $\mu_h(V_h)$ is compensated by increasing the processing requirement of that task* (Fig.16) *so that $\mu_h(\cdot) \sim \mu_m(\cdot)$ are kept to be even. By this modification, the fixed set of all tasks under consideration is slightly changed from $\mathcal{T}$ to $\overline{\mathcal{T}}$. But the partition $\mathcal{X} \cong \{V_j\}$ is kept to be $\vec{\alpha}$-simple and even as for $h \sim m$ (the value $L_{h\sim m}$ is unchanged). After this treatment, we decrease the percentage with which T_x belongs to V_h by ϵ, i.e. $\epsilon \cdot T_x$ is shifted from V_h to V_a. Then, under the condition that mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_h$ and the percentages with which each of tasks on boundaries among $C_1 \sim C_h$ is shared by corresponded $V_j s(P_j s)$ are all fixed as they are, $\mu_h(\cdot) \sim \mu_m(\cdot)$ are re-equalized $\vec{\alpha}$-simply for $\overline{\mathcal{T}}$ by adjusting $\sum_{j=1}^{h} \alpha_j$ (or $\sum_{k=h+1}^{m} \alpha_k$) and mutual ratios among $\alpha_{h+1}, \alpha_{h+2}, \ldots, \alpha_m$. (This is possible from induction hypothesis.) Assuming that it is realized at $\vec{\alpha} = (\alpha'_1, \alpha'_2, \ldots, \alpha'_m)$, let $(\vec{\alpha}', \delta', \mathcal{X}' \cong \{V'_j\}, L'_{h\sim m})$ denote this partition state. Last, task-fractions modified at first (if any) are restored to originals. The partition $\mathcal{X}' \cong \{V'_j\}$ for $\overline{\mathcal{T}}$ is also $\vec{\alpha}$-simple and even for $h \sim m$ for $\mathcal{T}$ (the value $L'_{h\sim m}$ is unchanged).*

The above `operation 2` has the following properties, P1, P2 and P3. (Their proofs are given in Appendix.)

[P1]. *When $\mu_h(\cdot) \sim \mu_m(\cdot)$ are re-equalized for $\overline{\mathcal{T}}$ and then for $\mathcal{T}$, the boundary between C_a and C_b never moves toward $/a/$ for any a, b ($1 \le a \le h$, $h+1 \le b \le m$). That is, it holds that*

$$\forall a \in \{1, \ldots, h-1, h\}, \quad \forall b \in \{h+1, h+2, \ldots, m\} \qquad \frac{\alpha_a}{\alpha_b} \le \frac{\alpha'_a}{\alpha'_b}.$$

[P2]. `Operation 2` *changes $L_{h\sim m}$ for $\mathcal{T}$ continuously and decreasingly with respect to $\epsilon (\in [0,1])$. That is, it holds that*

$$L_{h\sim m} - \epsilon \cdot \mu_h(T_x) \le L'_{h\sim m} < L_{h\sim m}.$$

[P3]. `Operation 2` *changes $\mu_{h-1}(V_{h-1})$ for $\mathcal{T}$ continuously and nondecreasingly with respect to $\epsilon (\in [0,1])$. That is, it holds that*

$$\mu_{h-1}(V_{h-1}) \le \mu_{h-1}(V'_{h-1}) \le \mu_{h-1}(V_{h-1}) + M \cdot \epsilon, \qquad M : \text{a positive constant.}$$

`Operation 2` with $\epsilon = 1$ shifts entire T_x to V_a. We repeat `Operation 2` until all of tasks on boundaries between C_a ($a = 1, 2, \ldots, h-1$) and C_b ($b = h, h+1, \ldots, m$) (including the tasks which newly locate on the boundaries while these operations) are shifted to $V_1 \sim V_{h-1}$. Let $(\vec{\alpha}^1, \delta^1, \mathcal{X})$ denote the partition state where all tasks on the boundaries are shifted out, and we again begin to apply `Operation 1` to this state. δ being increased from δ^1, the boundaries again pass a task at some value of δ, say δ^{11}, then `Operation 2` is applied, and so on.

$$(\vec{\alpha}^0, \delta^0) \xrightarrow{\texttt{Op.1}} (\vec{\alpha}^{00}, \delta^{00}) \underset{\texttt{Op.2}}{\Longrightarrow} (\vec{\alpha}^1, \delta^1) \xrightarrow{\texttt{Op.1}} (\vec{\alpha}^{11}, \delta^{11}) \underset{\texttt{Op.2}}{\Longrightarrow} \cdots \underset{\texttt{Op.2}}{\Longrightarrow} (\vec{\alpha}^e, \delta^e)$$

By P1 and definition of `Operation` 2 the above process terminates at some partition state $(\vec{\alpha}^e, \delta^e)$ $(\delta^e < 1)$ where any of $C_h, \ldots, C_m$ contains no task in it. Consequently we have

$$\begin{cases} L_{h\sim m} = \mu_h(V_h) = \mu_{h+1}(V_{h+1}) = \cdots = \mu_m(V_m) = 0 \\ \mu_{h-1}(V_{h-1}) \geq 0 . \end{cases}$$

P2 and P3 assert that $L_{h\sim m}$ and $\mu_{h-1}(V_{h-1})$ change monotonously as a partition state is changed from $(\vec{\alpha}^0, \delta^0)$ to $(\vec{\alpha}^e, \delta^e)$ (see Fig.17).

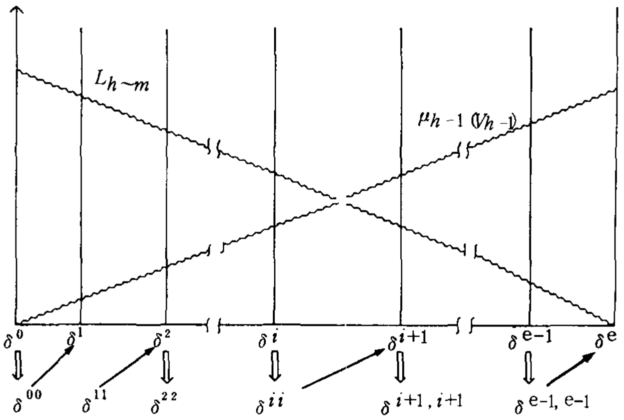


Fig.17 changes of $L_{h\sim m}$ and $\mu_{h-1}(V_{h-1})$

$L_{h\sim m}$ decreases monotonously, and $\mu_{h-1}(V_{h-1})$ changes nondecreasingly, therfore they cross in some interval, say $[(\vec{\alpha}^{ii}, \delta^{ii}), (\vec{\alpha}^{i+1}, \delta^{i+1})]$. Let $T_{x_1}, T_{x_2}, \ldots, T_{x_N}$ be tasks on boundaries (including those newly located) shifted from V_h to V_a $(1 \leq a \leq h-1)$ by `Operation` 2 in this interval, and let ϵ denote the shifted portion of the current task.

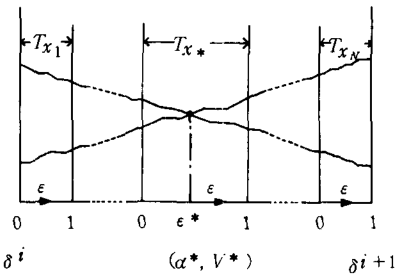


Fig.18 the crossing interval

P2 and P3 guarantee that $L_{h\sim m}$ and $\mu_{h-1}(V_{h-1})$ are continuous with respect to $\epsilon \in [0,1]$. Then there exist such T_{x_*} and ϵ^* that correspond to partition state $(\vec{\alpha}^*, \delta^*, \mathcal{X}^*)$ which realize

$$\mu_{h-1}(V_{h-1}) = L_{h\sim m} = \mu_h(V_h) = \mu_{h+1}(V_{h+1}) = \cdots = \mu_m(V_m) .$$

Here, $\vec{\alpha}^*$-simple partition keeps the fixed mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_{h-1}$ and the fixed percentages with which each of tasks on boundaries are shared. And it is easily seen that $\alpha_h^* \sim \alpha_m^* \neq 0$. This means the induction hypothesis holds for $l = h - 1$.

The above completes the mathematical induction, i.e. Lemma 1 holds for $l = m - 1, m - 2, \ldots, 1$. Therefore we have Theorem 1. $\square$

4.3 Another proof of Theorem 2

We prove Theorem 2 considering directed graphs associated with exchanges of task-fractions among processors, where a reduction to absurdity (proof by contradiction) is used.

proof : We suppose that some $\vec{\alpha}$-simple and even partition $\mathcal{X}^o = (x_{ij}) \cong \{V_1^o, V_2^o, \ldots, V_m^o\}$ does not give the minimum schedule for some task set $\mathcal{T}$. From this supposition we can derive a contradiction.

The optimum solution of (3.1) computed by, say, the simplex method determines the partition $\mathcal{X}^* \cong \{V_1^*, V_2^*, \ldots, V_m^*\}$ that gives the minimum schedule length. Here we consider to change $\mathcal{X}^o$ into $\mathcal{X}^*$ by shifting elements of each V_j^o (dividing them again if necessary) to the others. Let τ_{jk} $(1 \leq j, k \leq m)$ denote the set of task-fractions which are shifted from P_j (V_j^o) to P_k (V_k^*) when the partition is changed from $\mathcal{X}^o$ to $\mathcal{X}^*$. That is,

$$V_j^* = V_j^o - \sum_{k=1}^{m} \tau_{jk} + \sum_{k=1}^{m} \tau_{kj}.$$

Generally let $\boldsymbol{T} = (\tau_{jk})$ denote a collection of these m^2 sets of shifted task-fractions which can be applied to $\mathcal{X}^o \cong \{V_j^o\}$. And let $\#\boldsymbol{T}$ denote the number of non-empty entries of $\boldsymbol{T}$. We select from possibly plural $\mathcal{X}^*$s (there may be many optimum solutions when (3.1) degenerates), $\underline{\mathcal{X}}^*$, one of the partition that has the smallest $\#\boldsymbol{T}$ $(= \nu_0)$ among all that gives the minimum schedule length. Let $\underline{\boldsymbol{T}} = (\underline{\tau}_{jk})$ be the shifting of task-fractions which changes $\mathcal{X}^o$ into $\underline{\mathcal{X}}^*$.

Now by these assumptions of a reduction to absurdity we have

- length($\underline{\mathcal{X}}^*$) < length($\mathcal{X}^o$)
- $\#\underline{\boldsymbol{T}} = \nu_0$
- Any exchange of task-fractions $\boldsymbol{T}'$ (applicable to $\mathcal{X}^o$) s.t. $\#\boldsymbol{T}' < \nu_0$ can not change $\mathcal{X}^o$ into the partition that gives the schedule length less than or equal to length($\mathcal{X}^*$) (= length($\underline{\mathcal{X}}^*$)).

In the followings we consider the directed graph $G(\underline{\boldsymbol{T}})$ whose vertices are corresponded to processors $P_1, P_2, \ldots, P_m$ and which has the edge from P_j to P_k if and only if $\underline{\tau}_{jk} \neq \phi$.

When $\mathcal{X}^o$ is changed into $\underline{\mathcal{X}}^*$ by $\underline{\boldsymbol{T}}$, from every processor some task-fractions must be shifted to others. That is

$$\forall j, \qquad \exists k \quad s.t. \; \underline{\tau}_{jk} \neq \phi.$$

If otherwise, for some j, V_j^o is not decreased, then we have that $\mu_j(V_j^*) \geq \mu_j(V_j^o)$. Since the partition $\mathcal{X}^o$ is even, this contradicts that length($\underline{\mathcal{X}}^*$) < length($\mathcal{X}^o$). Consequently each vertex of the graph $G(\underline{\mathcal{T}})$ has one or more outgoing edges, and we can trace these edges in sequence. There are m finite vertices, so m or less operations of tracing necessarily lead to the same vertex visited before. In other words, there exists a following sequence of processors (a cycle of $G(\underline{\mathcal{T}})$),

$$P_{j_1}, P_{j_2}, \ldots, P_{j_e}, P_{j_{e+1}} \equiv P_{j_1}$$
$$\text{where} \quad \mathcal{T}_{j_s, j_{s+1}} \neq \phi \qquad s = 1, 2, \ldots, e \quad \text{(see Fig.19).}$$

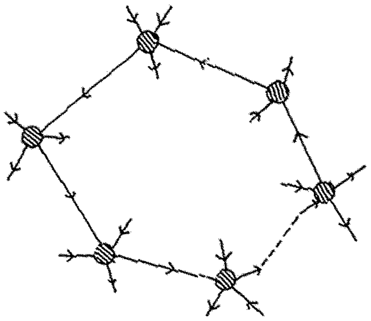


Fig.19 cycle of $G(\underline{\mathcal{T}})$

Here we apply the following `Operation 3` to this cycle (sequence of processors). To keep notations compact, processors are reindexed so that the index j_s on the cycle is changed to s ($s = 1, 2, \ldots, e$, where $e + 1 \equiv 1$).

```
Operation 3
    $\theta_1 \leftarrow 1$
    for $s := 2$ to $e$ do [
      if $\mu_s(\mathcal{T}_{s-1,s}(\theta_{s-1})) \le \mu_s(\mathcal{T}_{s,s+1})$
        then (* Fig.20 *)
          let $\theta_s \leftarrow \frac{\mu_s(\mathcal{T}_{s-1,s}(\theta_{s-1}))}{\mu_s(\mathcal{T}_{s,s+1})}$
        else (* Fig.21 *)
          let $\theta_s \leftarrow 1$;
          let $\rho \leftarrow \frac{\mu_s(\mathcal{T}_{s,s+1})}{\mu_s(\mathcal{T}_{s-1,s}(\theta_{s-1}))}$
          for $i := 1$ to $s - 1$ do [
            $\theta_i \leftarrow \theta_i \cdot \rho$
          ]
    ]
```

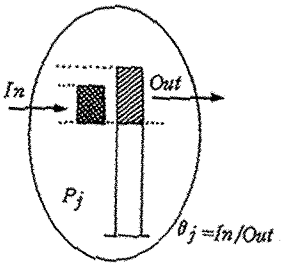


Fig.20 the case of $\mu_s(In) \le \mu_s(Out)$

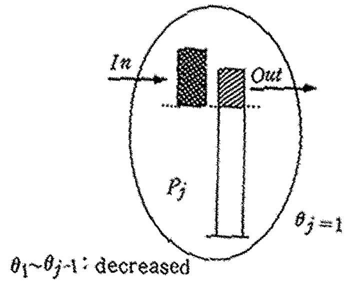


Fig.21 the case of $\mu_s(In) > \mu_s(Out)$

For the values $\theta_1, \theta_2, \ldots, \theta_e$ computed by `Operation 3`, we have the following Lemma 2, 3 and 4.

Lemma 2. *There exists such θ_s that equals to 1 among $\theta_1, \theta_2, \ldots, \theta_e$.*

proof of Lemma 2 : It is clear through the mathematical induction on e. $\square$

Lemma 3. *At processor P_s $(s = 2, 3, \ldots, e)$ it holds that*

$$\mu_s(\underline{\tau}_{s-1,s}(\theta_{s-1})) = \mu_s(\underline{\tau}_{s,s+1}(\theta_s)).$$

proof of Lemma 3 : Obvious from the description of `Operation 3` (see Fig.22). $\square$

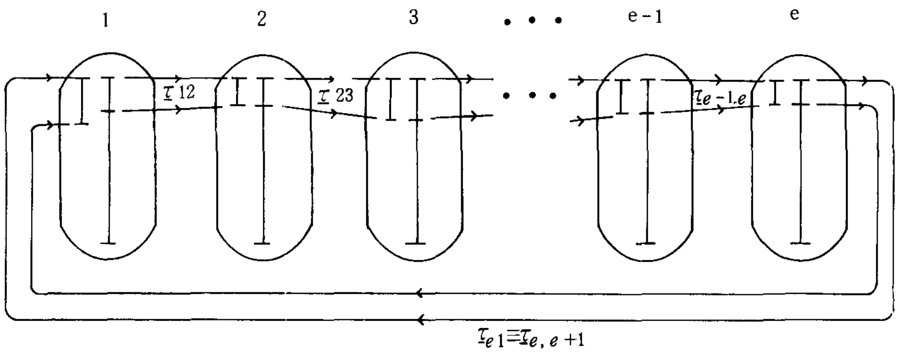

1
2
3
e-1
e
$\underline{\tau}_{12}$
$\underline{\tau}_{23}$
$\underline{\tau}_{e-1,e}$
$\underline{\tau}_{e1} \equiv \underline{\tau}_{e,e+1}$

Fig.22 sequence of similarly reduced $\underline{\tau}_{s,s+1}$ by `Operation 3`

Lemma 4. *At processor P_1 it holds that $\mu_1(\underline{\tau}_{12}(\theta_1)) \le \mu_1(\underline{\tau}_{e,1}(\theta_e))$.*

proof of Lemma 4 : Because the partition $\mathcal{X}^o$ is $\vec{\alpha}$-simple and $\underline{\tau}_{s,s+1}(\theta_s)$ is a subset of V_s^o, every element ω of $\underline{\tau}_{s,s+1}(\theta_s)$, satisfies the following relation (see Fig.23).

$$\frac{\mu_s(\omega)}{\mu_{s+1}(\omega)} \le \frac{\alpha_s}{\alpha_{s+1}}$$

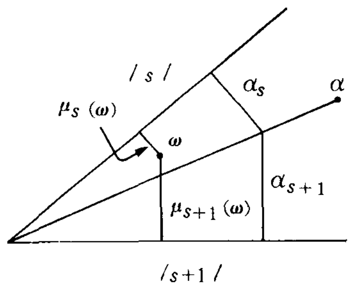


Fig.23 task-fraction that belongs to $\underline{\tau}_{s,s+1}$

Consequently it holds that

$$\frac{\mu_s(\underline{\tau}_{s,s+1}(\theta_s))}{\mu_{s+1}(\underline{\tau}_{s,s+1}(\theta_s))} \le \frac{\alpha_s}{\alpha_{s+1}}. \tag{4.12}$$

The Inequality of (4.12) is rewritten with $s = 1, 2, \ldots, e$ as

$$\begin{array}{ccccccc}
\mu_1(\underline{\tau}_{12}(\theta_1)) & / & \mu_2(\underline{\tau}_{12}(\theta_1)) & \le & \alpha_1 & / & \alpha_2 \\
\mu_2(\underline{\tau}_{23}(\theta_2)) & / & \mu_3(\underline{\tau}_{23}(\theta_2)) & \le & \alpha_2 & / & \alpha_3 \\
\cdot & / & \cdot & \le & \cdot & / & \cdot \\
\cdot & / & \cdot & \le & \cdot & / & \cdot \\
\cdot & / & \cdot & \le & \cdot & / & \cdot \\
\mu_{e-1}(\underline{\tau}_{e-1,e}(\theta_{e-1})) & / & \mu_e(\underline{\tau}_{e-1,e}(\theta_{e-1})) & \le & \alpha_{e-1} & / & \alpha_e \\
\mu_e(\underline{\tau}_{e,1}(\theta_e)) & / & \mu_1(\underline{\tau}_{e,1}(\theta_e)) & \le & \alpha_e & / & \alpha_1 \, .
\end{array}$$

Multiplying these inequalities, we have $\mu_1(\underline{\tau}_{12}(\theta_1))/\mu_1(\underline{\tau}_{e,1}(\theta_e)) \le 1$ by Lemma 3. That is, it holds that $\mu_1(\underline{\tau}_{12}(\theta_1)) \le \mu_1(\underline{\tau}_{e,1}(\theta_e))$. $\square$

Now we subtract $\{\underline{\tau}_{s,s+1}(\theta_s)\}$ $(s = 1, 2, \ldots, e)$ from the original $\underline{\boldsymbol{T}}$, and let $\underline{\boldsymbol{T}}^- = (\underline{\tau}^-_{jk})$ be the difference. More strictly,

$$\underline{\tau}^-_{jk} = \begin{cases} \underline{\tau}_{jk} & (j,k) \ne (s, s+1) \ (s = 1, 2, \ldots, e) \\ \underline{\tau}_{jk} - \underline{\tau}_{s,s+1}(\theta_s) & (j,k) = (s, s+1) \ (s = 1, 2, \ldots, e) . \end{cases}$$

The schedule length of the partition $\underline{\mathcal{X}}^-$ obtained by applying $\underline{\boldsymbol{T}}^-$ to the original $\mathcal{X}^o$ satisfies that length$(\underline{\mathcal{X}}^-) \le$ length$(\underline{\mathcal{X}}^*)$ by Lemma 3, 4. At the same time the edge corresponded to $\theta_{j_s} = 1$ (by Lemma 2) is deleted, so we have that $\#\underline{\boldsymbol{T}}^- < \nu_0$. These contradict the first assumptions. Therefore $\mathcal{X}^o$ must give the minimum schedule length, then Theorem 2 holds. $\square$

5. Application of Theorems

Most of approximate algorithms for the original (nonpreemptive) problem solve relaxed problems of type (3.1) repeatedly [7,9]. So it is needed to solve them as fast as possible. Theorem 1 and 2 give the possibility to obtain the solution of (3.1) faster than by such general methods as Simplex or Interior ones.

In the case of $m = 2$, the optimum value of $\vec{\alpha} \in \mathcal{A}_1$ can be found directly in $\mathcal{O}(n)$ time. This is implicitly used in [8]. When $m = 3$, the $\mathcal{O}(n \log n)$ algorithm which finds $\vec{\alpha}$-simple and almost even (even within certain error bound) partition of $\mathcal{T}$ is guided naturally by considering the process of the proof of Theorem 1. Using this procedure we can construct an approximate algorithm with worst case performance ratio 5/3 based on the idea of partial enumeration (essentially similar to [8]). It solves several hundred variants of (3.1), but takes no more than a second or so to produce the approximate solution for $n = 50$ on ordinary large scale computer.

For general m, direct search procedure of the optimum $\vec{\alpha}$, temporarily sketched as

```
step0 : let $\vec{\alpha} \leftarrow \left(\frac{1}{m}, \frac{1}{m}, \cdots, \frac{1}{m}\right)$   (* center of $\mathcal{A}_{m-1}$ *)
step1 : search the $\max_j \mu_j(V_j)$   (* $j_*$ gives maximum *)
step2 : let $\vec{\alpha} \leftarrow \left(\frac{\alpha_1}{1-\alpha_{j_*}+\delta}, \cdots, \frac{\alpha_{j_*-1}}{1-\alpha_{j_*}+\delta}, \alpha_{j_*} - \delta, \frac{\alpha_{j_*+1}}{1-\alpha_{j_*}+\delta}, \cdots, \frac{\alpha_m}{1-\alpha_{j_*}+\delta}\right)$
```

```
step3 : adjust $\delta$ and update task assignment
step4 : if schedule length is improvable
            then goto step1 else stop ,
```

seems to work similarly to (dual) Simplex method. But updating both $\vec{\alpha}$ and task assignment (i.e. exchange of basic variables) needs relatively small operations, so this procedure is expected to run faster than general methods.

Furthermore our Theorems can be used in order to prepare the initial basic solution (initial task assignment) for various methods.

6. Conclusion

By extending the study of the case $m = 2$, we introduced a new approach to the problem (3.1) which uses $m-1$ dimensional unit simplex $\mathcal{A}_{m-1}$. Theorems which interpret the optimum solution of (3.1) as the certain division of $\mathcal{A}_{m-1}$ i.e. the even and $\vec{\alpha}$-simple partition of $\mathcal{T}$ are interesting by themselves. The compact proof using Duality Theorem is convincible of the correctness of these theorems. On the other hand the elementary and lengthy proofs give helpful informations to design direct search algorithms of the optimum $\vec{\alpha}$ as stated in the previous section.

Numerical experiments on the approximate algorithm for $m = 3$ are under way, including the comparison with other methods . To complete the direct search algorithms for general m is the subject of following works.

Acknowledgements

The author wishes to thank Professor Masanori Fushimi and Professor Masao Iri of the University of Tokyo for their guidances, the proof using Duality Theorem in section 4.1 is suggested by Prof. Iri. The author also wishes to thank anonymous referees who read the first version of this paper with not a few errors and confusions and gave valuable comments and suggestions to him.

Appendix

The properties P1, P2 and P3 of `Operation 2` are proved.

At first we note the following fact (Lemma A1), next prove that inequalities of P2 are true for $\overline{T}$ (Lemma A2), then prove P1 $\sim$ P3 using these Lemmata.

Lemma A1. *In* `Operation 2`, *the $\vec{\alpha}$-simple re-equalization of $P_h \sim P_m$ for $\overline{T}$ after shifting of $\epsilon \cdot T_x$ can be realized by $\boldsymbol{T} = (\tau_{jk})$ such that $G(\boldsymbol{T})$ has no cycle. Here τ_{jk} ($1 \leq j, k \leq m$) denotes a set of task-fractions which are shifted from V_j to V'_k when the partition state for $\overline{T}$ is changed from $(\vec{\alpha}, \delta, \mathcal{X} \cong \{V_j\}, L_{h\sim m})$ to $(\vec{\alpha}', \delta', \mathcal{X}' \cong \{V'_j\}, L'_{h\sim m})$, and $G(\boldsymbol{T})$ denotes the directed graph whose vertices are corresponded to processors $P_1, P_2, \ldots, P_m$ and which has the edge from P_j to P_k if and only if $\tau_{jk} \neq \phi$.*

proof of Lemma A1 : Suppose that $G(\boldsymbol{T})$ has a cycle $P_{j_1}, P_{j_2}, \ldots, P_{j_e}, P_{j_{e+1}} \equiv P_{j_1}$. The existence of non-empty $\tau_{j_s, j_{s+1}}$ ($s = 1, 2, \ldots, e$) means that the boundary between C_{j_s} and $C_{j_{s+1}}$ moves toward $/j_s/$ or it does not move at all (in this case shifted task-fractions are on the boundary) when the partition state is changed from $\mathcal{X}$ to $\mathcal{X}'$ (see Fig. A.1). That is, along the cycle (for $s = 1, 2, \ldots, e$) it holds that

$$\frac{\alpha_{j_{s+1}}}{\alpha_{j_s}} \leq \frac{\alpha'_{j_{s+1}}}{\alpha'_{j_s}} \qquad \text{(where } \alpha_*, \alpha'_* > 0\text{)}.$$

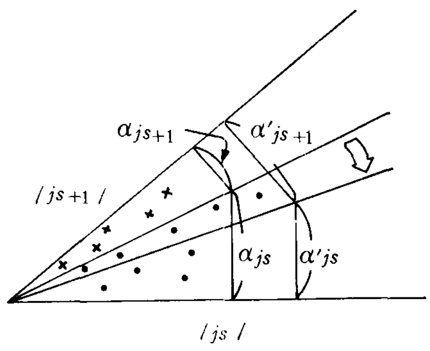


Fig.A.1 boundary between C_{j_s} and $C_{j_{s+1}}$

All of these inequalities hold, in fact, with equality, because, if some of them hold with true inequality then we have the contradiction that $1 < 1$ by multiplying them for $s = 1, 2, \ldots, e$. Therefore the boundary between C_{j_s} and $C_{j_{s+1}}$ is not moved, and all elements of $\tau_{j_s, j_{s+1}}$ are on this boundary (for $s = 1, 2, \ldots, e$). Then, we can eliminate at least one edge on each cycle by reversing some parts of $\tau_{j_s, j_{s+1}}$ s of the cycle in the same way as `Operation 3` of section 4.3. At this time it is easily seen that the resultant partition $\mathcal{X}''$ is kept to be $\vec{\alpha}$-simple and even (for $h, \ldots, m$). Repeating this operation leads to Lemma A1. □

Hereafter, without loss of generality, we assume that re-equalization in `Operation 2` is performed by the shifting of task-fractions $\boldsymbol{T} = (\tau_{jk})$ such that $G(\boldsymbol{T})$ has no cycle.

Lemma A2. *P1 whose T is replaced with $\overline{T}$ is true. That is, after $\mu_h(\cdot) \sim \mu_m(\cdot)$ are re-equalized for $\overline{T}$, it holds that*

$$L_{h\sim m} - \epsilon \cdot \mu_h(T_x) \leq L'_{h\sim m} < L_{h\sim m} .$$

proof of Lemma A2 : First, we show that $L'_{h\sim m} < L_{h\sim m}$. Since $G(\boldsymbol{T})$ has no cycle, there exists the following finite sequence of processors which starts from P_h and terminate at some processor P_{j_e} to which no task-fraction is shifted from any other processor during `Operation 2` (Fig. A.2).

$$P_{j_e}, P_{j_{e-1}}, \cdots, P_{j_1} \equiv P_h$$
$$\text{where } P_{j_s} \neq P_{j_t} \text{ iff. } s \neq t, \quad \tau_{j_s,j_{s-1}} \neq \phi \quad (s = e, e-1, \ldots, 2)$$
$$\text{and } \tau_{k,j_e} = \phi \quad (\forall k \in \{1, 2, \ldots, m\})$$

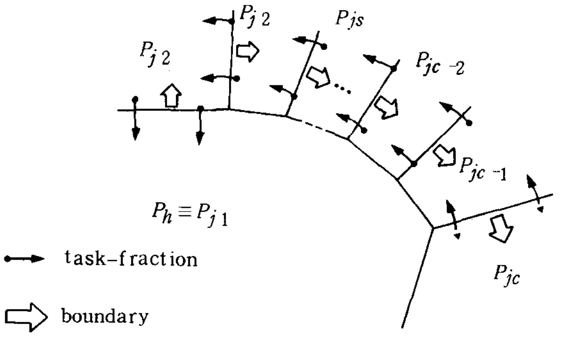


Fig.A.2 the sequence of processors coming into P_h

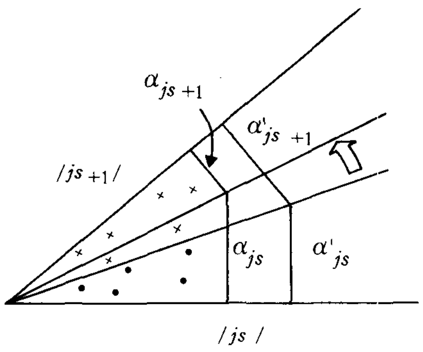


Fig.A.3 movement of a boundary

Here, if $j_e \in \{h, h+1, \ldots, m\}$ then it holds that

$$L'_{h\sim m} = \mu_{j_e}(V'_{j_e}) < \mu_{j_e}(V_{j_e}) = L_{h\sim m} ,$$

because V_{j_e} decreases strictly. Now supposing that $j_e \notin \{h, h+1, \ldots, m\}$, let j_c be the first index departing from $\{h, h+1, \ldots, m\}$. That is

$$j_1, j_2, \ldots, j_{c-1} \in \{h, h+1, \ldots, m\}$$
$$j_c \in \{1, 2, \ldots, h-1\} .$$

Since τ_{j_s+1,j_s} is not empty set, we have, as shown in proof of Lemma A1,

$$\frac{\alpha_{j_s}}{\alpha_{j_s+1}} \leq \frac{\alpha'_{j_s}}{\alpha'_{j_s+1}} \qquad s = 1, 2, \ldots, c-2 .$$

On the other hand, the partition $\mathcal{X}$ for $\overline{T}$ has no task-fractions on the boundary between C_a and C_b for any a, b $(1 \leq a \leq h,\ h+1 \leq b \leq m)$ which can be shifted from V_a to V_b (see Fig. A.4). Consequently $\tau_{j_c,j_{c-1}} \neq \phi$ implies that

$$\frac{\alpha_{j_c-1}}{\alpha_{j_c}} < \frac{\alpha'_{j_{c-1}}}{\alpha'_{j_c}} .$$

And, in Operation 2, mutual ratios among $\alpha_1, \alpha_2, \ldots, \alpha_h$ are fixed, so we have

$$\frac{\alpha_{j_c}}{\alpha_{j_1}} = \frac{\alpha'_{j_c}}{\alpha'_{j_1}} \qquad (\text{because} \quad j_1 = h, \quad 1 \le j_c \le h-1).$$

By multiplying all these (in)equalities we have the contradiction that $1 < 1$. Therefore j_e must belong to $\{h, h+1, \ldots, m\}$, and it holds that $L'_{h\sim m} < L_{h\sim m}$.

Next, we see that $L_{h\sim m} - \epsilon \cdot \mu_h(T_x) \le L'_{h\sim m}$. If no other task-fraction than T_x is shifted from P_h to other processors then this assertion is true. So, suppose that there exists some processor P_{j_2} to which task-fractions are shifted from P_h during Operation 2, i.e. $\tau_{h,j_2} \ne \phi$. j_2 satisfies that $h+1 \le j_2 \le m$ by definition of Operation 2. Now, by Lemma A1, we can consider the following finite sequence of processors which starts from $P_h \equiv P_{j_1}$ and terminates at some processor P_{j_e} from which no task-fraction is shifted to any other processor during Operation 2 (Fig. A.5).

$$\begin{aligned}&P_h \equiv P_{j_1}, P_{j_2}, \cdots, P_{j_e}\\ &\text{where } P_{j_s} \ne P_{j_t} \text{ iff. } s \ne t, \qquad \tau_{j_s, j_{s+1}} \ne \phi \quad (s = 1, 2, \ldots, e-1)\\ &\qquad \text{and} \quad \tau_{j_e, k} = \phi \quad (\forall k \in \{1, 2, \ldots, m\})\end{aligned}$$

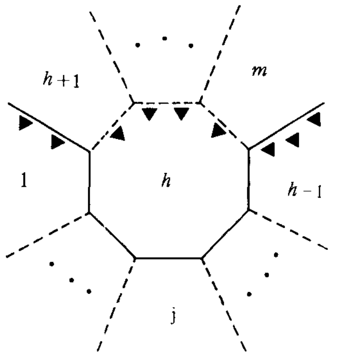


Fig.A.4 boundary condition on $\mathcal{T}$

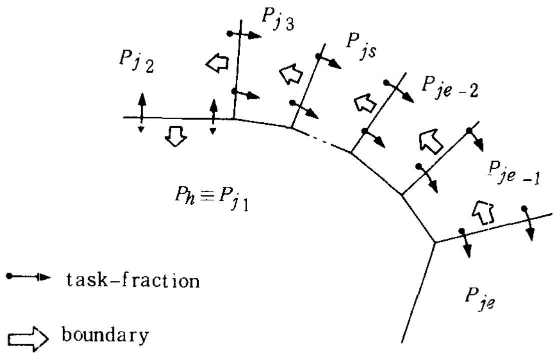


Fig.A.5 the sequence of processors starting from P_h

Here, if j_e is a member of $\{h, h+1, \ldots, m\}$ then we have the relation that $\mu_{j_e}(V'_{j_e}) \ge \mu_{j_e}(V_{j_e})$, which contradicts the relation that $L'_{h\sim m} < L_{h\sim m}$. Consequently we have $j_e \in \{1, 2, \ldots, h-1\}$. Now, we examine the movements of boundaries between C_{j_s} and $C_{j_{s+1}}$ for $s = 1, 2, \ldots, e$ (sub-index $e+1$ is interpreted as 1). The boundary between C_h and C_{j_2} must move toward $/h/$, that is

$$\frac{\alpha_{j_2}}{\alpha_h} < \frac{\alpha'_{j_2}}{\alpha'_h},$$

because $\mathcal{T}$ has no task-fraction on the boundary between C_h and C_b for any b $(1 \le b \le m)$ which can be shifted from V_h to V_b (see Fig.A.4). About other boundaries, in the same way as the former case, we have

$$\frac{\alpha_{j_s+1}}{\alpha_{j_s}} \le \frac{\alpha'_{j_s+1}}{\alpha'_{j_s}} \qquad (s = 2, 3, \ldots, e-1),$$

$$\frac{\alpha_h}{\alpha_{j_e}} = \frac{\alpha'_h}{\alpha'_{j_e}} \qquad (\text{because } 1 \le j_e \le h-1).$$

By multiplying these (in)equalities we again have the contradiction that $1 < 1$. Therefore no task-fraction can be shifted from V_h to any other V_k $(k \in \{1, 2, \ldots, m\})$ except for $\epsilon \cdot T_x$ to some V_a $(1 \le a \le h-1)$. That is, it holds that $L_{h\sim m} - \epsilon \cdot \mu_h(T_x) \le L'_{h\sim m}$. □

proof of P1 : Suppose that for some a, b $(1 \le a \le h, h+1 \le b \le m)$ the boundary between $\mathcal{C}_a$ and $\mathcal{C}_b$ moves toward $/a/$, that is

$$\frac{\alpha_a}{\alpha_b} > \frac{\alpha'_a}{\alpha'_b}.$$

Since mutual ratios among α_1, α_2, ..., α_h are fixed during `Operation 2`, we have

$$\forall j \in \{1, 2, \ldots, h\} \qquad \frac{\alpha_j}{\alpha_b} > \frac{\alpha'_j}{\alpha'_b} \qquad \left(\text{because } \frac{\alpha_a}{\alpha_j} = \frac{\alpha'_a}{\alpha'_j} = \text{constant}\right).$$

Consequently, for any j $(\in \{1, 2, \ldots, h\})$, $\tau_{bj} = \phi$. But, considering that $\mu_b(V'_b) < \mu_b(V_b)$, there exists some k $(\in \{h+1, h+2, \ldots, m\})$ such that $\tau_{bk} \ne \phi$. Then, by Lemma A1, we can find the following finite sequence of processors which starts from $P_b \equiv P_{j_1}$ and terminates at some processor P_{j_e} from which no task-fraction is shifted to any other processor during `Operation 2`.

$$\begin{aligned}
&P_b \equiv P_{j_1},\ P_{j_2}, \cdots, P_{j_e} \\
&\text{where } P_{j_s} \ne P_{j_t} \text{ iff. } s \ne t, \qquad \tau_{j_s, j_s+1} \ne \phi \quad (s = 1, 2, \ldots, e-1) \\
&\qquad \text{and} \quad \tau_{j_e, k} = \phi \quad (\forall k \in \{1, 2, \ldots, m\})
\end{aligned}$$

The same discussions about the above sequence as in the latter half of proof of Lemma A2 conclude that j_e belongs to $\{1, 2, \ldots, h-1\}$. Multiplication of (in)equalities representing movements of the boundaries between every two consecutives of $\mathcal{C}_a$, $\mathcal{C}_b \equiv \mathcal{C}_{j_1}$, $\mathcal{C}_{j_2}$, $\cdots$, $\mathcal{C}_{j_{e-1}}$, $\mathcal{C}_{j_e}$, $\mathcal{C}_a$ bring the contradiction that $1 < 1$, hence P1 holds. □

proof of P2 : P1 assures that the partition $\mathcal{X}' \cong \{V'_j\}$ of $\overline{\mathcal{T}}$ is kept to be even for $h \sim m$ and $\vec{\alpha}$-simple when the task set is restored to original $\mathcal{T}$. (the value of $L_{h\sim m}$ is also unchanged). Consequently Lemma A2 holds for the original task set $\mathcal{T}$. In other words `Operation 2` changes the partition for $\mathcal{T}$ from $\mathcal{X}$ to $\mathcal{X}'$. □

$$\begin{tikzcd}
\mathcal{T}/(\mathcal{X}, L_{h\sim m}) \arrow[d, "\text{modify task-fractions on boundaries}"'] & & \mathcal{T}/(\mathcal{X}', L'_{h\sim m}) \\
\overline{\mathcal{T}}/(\mathcal{X}, L_{h\sim m}) \arrow[rr, "\text{re-equalization}", "\text{after } \epsilon\cdot T_x \text{ is shifted}"'] & & \overline{\mathcal{T}}/(\mathcal{X}', L'_{h\sim m}) \arrow[u, "\text{restore modified task-fractions}"']
\end{tikzcd}$$

proof of P3 : P1 and definition of `Operation` 2 show that no task-fraction is shifted out from V_{h-1} during `Operation` 2. From this fact it is obvious that $\mu_{h-1}(V_{h-1}) \leq \mu_{h-1}(V'_{h-1})$. So, it suffices to estimate the processing requirements of task-fractions which are shifted into V_{h-1}. We introduce following notations.

Q : $\{1, 2, \ldots, h-1\}$; set of indeces of processors.

R : $\{h, h+1, \ldots, m\}$; set of indeces of processors.

ρ : $\max_i \max_{j,k} \frac{\mu_{ik}}{\mu_{ij}}$; maximum (over all tasks) of maximum ratio of μ s.

Here let I_b/O_b denote the sets of all incoming/outgoing task-fractions into/from V_b ($b \in R$). Since P1 assures that ther is no task-fraction shifted into V_b from V_a during `Operation` 2 ($a \in Q$, $b \in R$), they are written as

$$I_b = \sum_{j \in R} \tau_{jb} \qquad b \in R$$

$$O_b = \sum_{j \in R} \tau_{bj} + \sum_{k \in Q} \tau_{bk} \qquad b \in R.$$

Measuring both sides of these equalities by $\mu_{h-1}(\cdot)$ (i.e. processing requirements on processor P_{h-1}), we have

$$\mu_{h-1}(I_b) = \sum_{j \in R} \mu_{h-1}(\tau_{jb}) \qquad b \in R$$

$$\mu_{h-1}(O_b) = \sum_{j \in R} \mu_{h-1}(\tau_{bj}) + \sum_{k \in Q} \mu_{h-1}(\tau_{bk}) \qquad b \in R.$$

Summing up respective equalities for all $b \in R$ and taking the difference between them, we have

$$\text{(a.1)} \quad \sum_{b \in R} \mu_{h-1}(O_b) - \sum_{b \in R} \mu_{h-1}(I_b)$$
$$= \sum_{b \in R} \sum_{j \in R} \mu_{h-1}(\tau_{bj}) + \sum_{b \in R} \sum_{k \in Q} \mu_{h-1}(\tau_{bk}) - \sum_{b \in R} \sum_{j \in R} \mu_{h-1}(\tau_{jb})$$
$$= \sum_{b \in R} \sum_{k \in Q} \mu_{h-1}(\tau_{bk}).$$

Here, by Lemma A2 we have

$$\text{(a.2)} \qquad \mu_b(O_b) - \mu_b(I_b) = L_{h\sim m} - L'_{h\sim m} \leq \epsilon \cdot \mu_h(T_x) \qquad \forall b \in R.$$

From this inequality we have

$$\text{(a.3)} \qquad \begin{cases} \mu_b(I_b) = o(\epsilon) & \forall b \in R \\ \mu_b(O_b) = o(\epsilon) & \forall b \in R. \end{cases}$$

Because if $\mu_b(I_b)$ or $\mu_b(O_b)$ is $o(1)$ for some b then we have the contradiction as follows. Suppose $\mu_{b_0}(I_{b_0}) = o(1)$ (by (a.2) $\mu_b(O_b) = o(1)$ is equivalent to $\mu_b(I_b) = o(1)$). This means that for some $b_1 \in R$, $\mu_{b_0}(\tau_{b_1,b_0}) = o(1)$. Here, by definition of ρ, it holds

$$\frac{1}{\rho}\mu_{b_0}(\tau_{b_1,b_0}) \leq \mu_{b_1}(\tau_{b_1,b_0}) \leq \rho\mu_{b_0}(\tau_{b_1,b_0}).$$

Consequently, $\mu_{b_1}(O_{b_1})$ is also $o(1)$, and it follows that $\mu_{b_1}(I_{b_1}) = o(1)$. Again starting from b_1 we can find b_2 s.t. $\mu_{b_2}(I_{b_2}) = o(1)$ and $\tau_{b_2,b_1} \neq \phi$. This process can be repeated endlessly as b_0, b_1, ..., that is, $G(\boldsymbol{T})$ has a cycle. But this contradicts the assumption that $G(\boldsymbol{T})$ has no cycle.

Obviously

$$\mu_{h-1}(V'_{h-1}) - \mu_{h-1}(V_{h-1}) \leq \sum_{b\in R}\sum_{k\in Q} \mu_{h-1}(\tau_{bk}).$$

From (a.1) we have

$$\text{(a.4)} \qquad \begin{aligned} \mu_{h-1}(V'_{h-1}) - \mu_{h-1}(V_{h-1}) &\leq \sum_{b\in R}\sum_{k\in Q} \mu_{h-1}(\tau_{bk}) \\ &\leq \sum_{b\in R} \mu_{h-1}(O_b). \end{aligned}$$

By (a.3) and definition of ρ, we have (for certain constant C_b)

$$\mu_{h-1}(O_b) \leq \rho\mu_b(O_b) \leq \epsilon C_b \qquad \forall b \in R.$$

Substituting these inequalities into (a.4), we have

$$\begin{aligned} \mu_{h-1}(V'_{h-1}) - \mu_{h-1}(V_{h-1}) &\leq \sum_{b\in R} \mu_{h-1}(O_b) \\ &\leq \sum_{b\in R} \epsilon C_b \\ &\leq \epsilon \cdot M, \end{aligned}$$

where $M = \sum_{b\in R} C_b$ is constant. $\square$

References

[1] Bruno, J., Coffman, E. G., Jr., and Sethi, R.: Scheduling Independent Tasks to Reduce Mean Finishing Time. *Comm. ACM* Vol.17, No.7 (1974), 382–387.

[2] Davis, E., and Jaffe, J. M.: Algorithms for Scheduling Tasks on Unrelated Processors. *J.ACM* Vol.28, No.4 (1981), 721–736.

[3] Garey, M. R., and Johnson, D. S.: *Computers and Intractability — A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.

[4] Horowitz, E., and Sahni, S.: Exact and Approximate Algorithms for Scheduling Nonidentical Processors. *J.ACM* Vol.23, No.2 (1976), 317–327.

[5] Ibarra, O. H., and Kim, C. E.: Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *J.ACM* Vol.24, No.2 (1977), 280–289.

[6] Iri, M.: *Linear Programming Methods* (in Japanese). Kyouritu, Tokyo, 1986.

[7] Lenstra, J. K., Shmoys, D. B., and Tardos, E.: Approximation Algorithms for Scheduling Unrelated Parallel Machines. *Report of Center for Mathematics and Computer Science* (Netherlands) OS-R8714, September, 1987.

[8] Numata, K.: An Approximate Algorithm for Scheduling Independent Tasks on Unrelated Processors (in Japanese). *Proceedings of the 1984th Fall Conference of the Operations Research Society of Japan*, 77–78.

[9] Numata, K.: Approximate and Exact Algorithms for Scheduling Independent Tasks on Unrelated Processors. *Journal of Operations Research Society of Japan*, Vol.31 No.1 (1988), 61–80.

[10] Potts, C. N.: Analysis of a Linear Programming Heuristic for Scheduling Unrelated Parallel Machines. *Discrete Appl. Math.* Vol.10 (1985), 155–164.

[11] Tamura, I.: *Topology (Iwanami Conpendium 276)* (in Japanese). Iwanami, Tokyo, 1972.

Kazumiti NUMATA: Department of Computer Science and Information Mathematics,
The University of Electro-Communications,
1-5-1 Chofu-ga-oka, Chofu-shi,
Tokyo, 182, Japan.