Minimizing Flow Time on Parallel Identical Processors with Variable Unit Processing Time

I. MEILIJSON and A. TAMIR
Tel-Aviv University, Tel-Aviv, Israel

(Received September 1981; accepted September 1982)

We show that minimizing total flow time on parallel identical processors with nonincreasing unit processing time is achieved with shortest processing time (SPT) scheduling: Sequence the tasks in an order of nondecreasing lengths. Following this order and starting with the shortest task, process any task by the first processor that becomes available. If the unit processing time is allowed to be increasing, e.g. quadratically, the problem becomes NP-hard.

ONE of the elementary results in scheduling theory refers to the following deterministic problem. A set of tasks $N = \{1, 2, \cdots, n\}$ specified by positive numbers $\{l_1, l_2, \cdots, l_n\}$ is to be processed on m identical machines. Each task must be processed by exactly one of the machines, preemption is not allowed, and the objective is to minimize the total flow (finishing) time. The number l_i is the length or work content of task i; that is, the number of units of the resource (e.g., lengths of iron bars) processed by the machines. If the machines are identical, and have a constant unit processing time μ (i.e., each unit of the resource requires μ time units to process), the schedule minimizing the total flow time is obtained as follows (Baker [1974, p. 119]): Sequence the tasks in an order of nondecreasing lengths. Following this order and starting with the shortest task, process any task by the first machine that becomes available. This processing scheme is called SPT (shortest processing time) scheduling.

The purpose of this work is to prove that the SPT schedule remains optimal even when the unit processing time μ of each one of the identical machines is any nonincreasing function of the total length of the resource that the machine has already processed. This processing time assumption could model, for example, situations in which a learning process determines the performance of the machine operators. Performance improves with experience.

We also show that if the unit processing time is allowed to be increasing, the problem becomes NP-hard.

Subject classification: 581 processors with variable processing rate.

To facilitate the discussion, suppose that the entire process starts at time $t = 0$. The machines are identical and each is associated with a positive nonincreasing, integrable unit processing time function $\mu(l)$. The processing time of a task of length l on a machine that has already processed tasks of total length L is $\int_L^{L+l} \mu(x)dx$. In minimizing the total flow (i.e., the sum of flow times), we may assume that an optimal schedule does not allow machines to insert idle times between consecutive tasks. Moreover, tasks assigned to the same machine are processed in order of nondecreasing lengths. In particular, if some machine is assigned the set of tasks $\{i_1, i_2, \cdots, i_k\}$, satisfying $l_{i_1} \leq l_{i_2} \leq \cdots \leq l_{i_k}$, the total flow on this machine is $\sum_{j=1}^{k} g(\sum_{s=1}^{j} l_{i_s})$,

where
$$g(l) = \int_0^l \mu(x)dx \tag{1}$$

is the clock time required by one machine to process l units of work content, and $(dg(l)/dl) = \mu(l)$. (Often, instead of working with $g(l)$ directly, we utilize the function $g^{-1}(t)$ = work content completed in t units of time by a variable speed processor. In this case, $\lambda(t) \equiv (dg^{-1}(t)/d(t)$ is usually called the processing rate λ and μ are related by the identity $\lambda(g(l)) = 1/\mu(l)$. Hence, nondecreasing $\lambda(t)$ corresponds to nonincreasing $\mu(l)$, as required above.) Note that the monotonicity of μ implies the concavity of $g(l)$. The total flow for the entire system is obtained by summing the flows over all the machines.

THEOREM 1. *Let $N = \{1, 2, \cdots, n\}$ be a set of tasks with lengths $\{l_1, l_2, \cdots, l_n\}$, respectively. Also, let $M = \{1, 2, \cdots, m\}$ be a set of identical machines having a positive, nonincreasing, and integrable unit processing time function $\mu(l)$. Then the SPT schedule minimizes the total flow of N on M.*

Proof. Suppose that $l_1 \leqq l_2 \leqq \cdots \leqq l_n$. Since all the machines have the same processing time function $\mu(l)$, the SPT schedule is implemented as follows: assign task i to machine $i(\text{mod} m)$ (machine m is now labeled as machine 0), and sequence the tasks on each machine in order of increasing task indices.

We first prove the result for $m = 2$, and then use the proof to validate the statement for a general m. (The case $m = 1$ is clearly valid even when $\mu(l)$ is any positive integrable function.)

We identify the different schedules of the tasks $\{1, 2, \cdots, n\}$ on the two machines as follows. Let $\pi = (\pi_1, \pi_2, \cdots, \pi_{n+1})$ denote a permutation of the vector $(1, 2, \cdots, n, n + 1)$. For our discussion, it is convenient to let π_i denote that coordinate of $(1, 2, \cdots, n + 1)$ that is mapped into the ith coordinate of π. Given π, if $n + 1 = \pi_k$, let tasks $(\pi_1, \pi_2, \cdots, \pi_{k-1})$ be processed on the first machine starting with π_1, then π_2 and so on. The remaining tasks $(\pi_{k+1}, \pi_{k+2}, \cdots, \pi_{n+1})$ are processed on the second machine in the reverse order: π_{n+1} starts first, π_n follows and so on. Define $l_{n+1} = l_1 + l_2 + \cdots + l_n$. Then $l_1 \le l_2 \le \cdots \le l_n \le l_{n+1}$. Also, define $L_{\pi_i} = \sum_{j=1}^{i} l_{\pi_j}$ for $1 \le i \le n + 1$. The completion time of task π_i is $g(L_{\pi_i})$ if $1 \le i \le k - 1$ and $g(L_{\pi_{n+1}} - L_{\pi_{i-1}})$ if $k + 1 \le i \le n + 1$. Therefore, the total flow of tasks $(1, 2, \cdots, n)$, while using the schedule induced by π, is

$$F(\pi) = \textstyle\sum_{i=1}^{k-1} g(L_{\pi_i}) + \sum_{i=k+1}^{n+1} g(L_{\pi_{n+1}} - L_{\pi_{i-1}}).$$

To obtain an expression that does not explicitly depend on the index k for which $n + 1 = \pi_k$, let

$$f(L) = \begin{cases} g(L) & \text{if } 0 \le L \le l_{n+1} \\ g(2l_{n+1} - L) & \text{if } l_{n+1} \le L \le 2l_{n+1}. \end{cases} \tag{2}$$

Using the relations $L_{\pi_{n+1}} = 2l_{n+1}$ and $l_{\pi_k} = l_{n+1}$, we easily verify that $L_{\pi_i} \le l_{n+1}$ for $1 \le i \le k - 1$ and $l_{n+1} \le L_{\pi_i} \le 2l_{n+1}$ for $k \le i \le n + 1$. Thus, applying (2) yields

$$F(\pi) = \textstyle\sum_{i=1}^{k-1} f(L_{\pi_i}) + \sum_{i=k}^{n} f(L_{\pi_i}).$$

Finally, observing that $f(L_{\pi_{n+1}}) = f(2l_{n+1}) = 0$, we obtain

$$F(\pi) = \textstyle\sum_{i=1}^{n+1} f(L_{\pi_i}) = \sum_{i=1}^{n+1} f(\sum_{j=1}^{i} l_{\pi_j}).$$

Minimizing the total flow of tasks $(1, 2, \cdots, n)$ corresponds to finding a permutation π^* for which $F(\pi)$ is minimized.

To prove that π^* is the permutation defining the SPT schedule, we use the results of Konheim and Meilijson [1980]. For the sake of completeness, we state and prove their relevant results in the Appendix. The function f, defined by (2), is concave and symmetric on $[0, 2l_{n+1}]$ with $f(0) = f(2l_{n+1}) = 0$. Therefore, Theorem A.1 in the Appendix, applied for the case $N = n + 1$, ensures that $F(\pi)$ is minimized when π^* is a greedy permutation of $(1, 2, \cdots, n + 1)$ in the space defined in the Appendix. Since $l_1 \le l_2 \le \cdots \le l_n \le l_{n+1}$, π^* induces the SPT schedule on the tasks $\{1, 2, \cdots, n\}$ and the proof for $m = 2$ is complete.

We now turn to the general case. Assume that the SPT scheduling is nonoptimal. Consider the set of optimal schedules that sequence on each machine according to the rule of smaller indexed task first. (This set is nonempty since we are assuming $l_1 \le l_2 \le \cdots \le l_n$.) Among those schedules, choose one for which the smallest indexed task i that is not assigned to machine $i(\text{mod } m)$ is maximum. Let that task be task k, and suppose that task k is assigned to machine u, $u \ne v = k(\text{mod } m)$. Let S_u and S_v be the sets of tasks assigned to machines u and v, respectively. Apply the optimality result of the SPT schedule to machines u and v

and the set of tasks $S_u \cup S_v$. The tasks in $S_u \cup S_v$ can be rescheduled on u and v without affecting the optimality of the given schedule on the m machines, and while ensuring that each task i, $i \le k$, is assigned to machine $i(\text{mod } m)$. This conclusion contradicts the maximality of k and completes the proof.

Simple examples, involving only two machines, demonstrate that Theorem 1 fails if the function $\mu(l)$ is not nonincreasing. In fact, we will next show that scheduling tasks on two machines with an increasing quadratic unit processing time function is *NP*-complete. An increasing unit processing time function may reflect fatigue phenomena.

Suppose, for example, that $\mu(l) = 3\,l^2$, and, therefore, $g(l)$, defined by (1), is given by $g(l) = l^3$. Let π denote a permutation of $\{1, 2, \cdots, n\}$. The objective is to find π minimizing the total flow, $\bar{F}(\pi)$, on the two machines,

$$\bar{F}(\pi) = \text{Min}_{1 \le k \le n} \left(\textstyle\sum_{j=1}^{k} \left(\sum_{i=1}^{j} l_{\pi_i}\right)^3 + \sum_{j=k+1}^{n} \left(\sum_{i=k+1}^{j} l_{\pi_i}\right)^3\right). \quad (3)$$

In this formulation, $\pi_i = j$ means that task j is the ith $((i - k)$th) task to be processed by the first (second) machine if $i \leqq k$ $(i > k)$.

THEOREM 2. *Given positive integers* $\{l_1, \cdots, l_n\}$ *and an integer* L, *the problem of determining whether there exists a permutation* π *with* $\bar{F}(\pi) \leqq L$ *is NP-complete.*

Proof. The problem is clearly in NP. We reduce the partition problem (see Garey and Johnson [1979]) to our scheduling problem. Given positive integers $\{a_1, \cdots, a_n\}$, the partition problem is defined as determining whether there exists a subset $S \subset \{1, \cdots, n\}$, called a partition, such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$.

Given the positive integers $\{a_1, a_2, \cdots, a_n\}$ with $A = \sum_{i=1}^{n} a_i$, define a set of $n + 2$ tasks, $\{1, 2, \cdots, n + 2\}$, as follows. For $1 \le i \le n$, let the length of task i be $l_i = a_i$, and $l_{n+1} = l_{n+2} = M$. M is chosen to be "very large," for example $M = A^{10}$.

We claim there exists a partition of $\{a_1, \cdots, a_n\}$ if, and only if, there exists a permutation π of the tasks $\{1, 2, \cdots, n + 2\}$ with $\bar{F}(\pi) \le 2(M + (A/2))^3 + M$. Suppose that there exists a partition $S \subset \{1, 2, \cdots, n\}$. Assign the tasks in $S \cup \{n + 1\}$ to one machine and the remaining tasks to the second machine. Since each machine (optimally) processes tasks in order of nondecreasing lengths, tasks $(n + 1)$ and $(n + 2)$ are to be processed last, yielding a flow of $(M + (A/2))^3$ for each of them. The flow of each task i, $1 \le i \le n$, is bounded by A^3. Thus, the total flow is at most $2(M + (A/2))^3 + nA^3 \le 2(M + (A/2))^3 + M$.

Next, suppose that there is no partition of $\{a_1, \cdots, a_n\}$. We will show that there is no schedule with total flow of at most $2(M + (A/2))^3 + M$. First, if there were such a schedule, it would not assign tasks $(n + 1)$ and

$(n + 2)$ to the same machine. (Otherwise, the sum of their flows would exceed $(M + M)^3 = 8M^3$.) The total flow of tasks $(n + 1)$ and $(n + 2)$ will then be $(M + A_1)^3$ and $(M + A_2)^3$, respectively, with $A_1 + A_2 = A$. Furthermore, since there is no partition, $A_1 = (A/2) + \alpha$ and $A_2 = (A/2) - \alpha$ with $\alpha \geq 1/2$. Therefore, $(M + A_1)^3 + (M + A_2)^3 = 2(M + A/2)^3 + 6\alpha^2(M + (A/2)) > 2(M + (A/2))^3 + M$, and there is no schedule with total flow of at most $2(M + (A/2))^3 + M$.

Several concluding comments are in order. First, observe that our scheduling model is equivalent to the following model. Given the tasks specified by $\{l_1, \cdots, l_n\}$, suppose that the machines are identical and have a constant unit processing time, normalized to one time unit per unit of work content. The tasks are now associated with a monotone cost function, $g(l)$, which is the cost incurred by a task if it completes its service at time l. The objective is to find a schedule minimizing the total cost incurred. Our results are, therefore, applicable to this latter model as well.

Next we note that the optimality of the SPT scheduling in Theorem 1 is not violated even under some mild priority restrictions. For example, suppose that certain tasks must start at time zero and be processed with no interruptions. The optimal scheduling procedure assigns each such task to a different machine, and then proceeds with the remaining tasks using SPT scheduling. The proof of optimality is similar to that of Theorem 1, and is, again, based on the results in Konheim and Meilijson.

APPENDIX

The following results, proved by Konheim and Meilijson, are presented here for the sake of completeness.

Let f be a symmetric concave function of one real variable (with a vertical axis of symmetry somewhere in the plane), $x = (x_1, x_2, \cdots, x_N)$ be a vector of positive real numbers, $S = x_1 + x_2 + \cdots + x_N$, and let α be a real number. Let $\pi = (\pi_1, \cdots, \pi_N)$ be a permutation of the vector $(1, 2, \cdots, N)$. (π_i is used to denote that coordinate of $(1, 2, \cdots, N)$ that is mapped into the ith coordinate of π).

For each permutation π of $(1, 2, \cdots, N)$, let

$$F(\pi) = f(\alpha) + \textstyle\sum_{i=1}^{N} f(\alpha + \sum_{j=1}^{i} x_{\pi_j}). \tag{A.1}$$

Suppose that $x_1 \leq x_2 \leq \cdots \leq x_N$. Define a permutation π to be greedy (on the triple (f, x, α)) recursively as follows:

$$\begin{aligned}
&\text{if } f(\alpha) < f(\alpha + S), && \text{set } \pi_1 = 1,\\
&\text{if } f(\alpha) = f(\alpha + S), && \text{set } \pi_1 = 1 \text{ or } \pi_N = 1 \text{ and}\\
&\text{if } f(\alpha) > f(\alpha + S), && \text{set } \pi_N = 1.
\end{aligned}$$

Having defined either the first or the last coordinate of π (i.e., π_1 or π_N), consider the reduced vector $x' = (x_2, x_3, \cdots, x_N)$ and the modified value of α

$$\alpha' = \begin{cases} \alpha + x_1 & \text{if} \quad \pi_1 = 1 \\ \alpha & \text{if} \quad \pi_N = 1 \end{cases}$$

and applying the rule specifying π_1 or π_N to the reduced triple (f, α, x'). The remaining values of π are determined recursively by this procedure. To be more specific, suppose that $f(\alpha) = f(\alpha + S)$. The following are greedy permutations.

$$\pi^* = \begin{cases} (1, 3, 5, \cdots, N-1, N, N-2, \cdots, 6, 4, 2) \\ \quad \text{or } (2, 4, 6, \cdots, N-2, N, N-1, \cdots, 5, 3, 1), \text{ if } N \text{ is even} \\ (1, 3, 5, \cdots, N-2, N, N-1, \cdots, 6, 4, 2) \\ \quad \text{or } (2, 4, 6, \cdots, N-1, N, N-2, \cdots, 5, 3, 1), \text{ if } N \text{ is odd.} \end{cases}$$

Clearly, $F(\pi)$ is the same for all greedy permutations.

THEOREM A.1. *The function $F(\pi)$, given by (A.1), defined on the set of all permutations, attains its minimum at the greedy permutations.*

For ease of exposition, consider f to describe the roof of a house (see Figure 1) with a one dimensional floor and rooms having preassigned lengths $\{x_i\}$. Theorem A.1 claims that the greedy builder, who minimizes at every stage of the construction the height of the current wall being built, makes the sum of the heights of the walls actually minimal.

LEMMA A.1. *For fixed x and f (symmetric and concave), let $V(\alpha)$ be the value of $F(\pi)$ for greedy permutations π, as a function of the left endpoint of the house. Then $V(\alpha)$ is continuous. If $f(\alpha) < f(\alpha + S)$, V is monotone nondecreasing on the interval $(-\infty, \alpha']$, where $\alpha' > \alpha$, and $f(\alpha') = f(\alpha' + S)$. If $f(\alpha) > f(\alpha + S)$, V is monotone nonincreasing on the interval $[\alpha'', \infty)$ where $\alpha'' < \alpha$ and $f(\alpha'') = f(\alpha'' + S)$. Moreover, if $f(\alpha) = f(\alpha + S)$, α is a maximum point of V.*

Proof. We first prove the continuity of the function $V(\alpha)$. Let α_1 and α_2 denote the leftmost maximum and rightmost maximum of f, respectively. In particular, each x such that $\alpha_1 \leq x \leq \alpha_2$ is a maximum point of f. It is clear that V is continuous at all points α at which the greedy permutation does not change, since the heights of the walls vary continuously. (For example, V is continuous for $\alpha \geq \alpha_2$ or $\alpha \leq \alpha_1 - S$.) Consider a point α at which the greedy permutation changes. Such an α is reached, when "sliding" the house to the right, whenever two walls become of the same height. At any such point, reverse the order of the rooms in the subhouse between these two walls. (If more than two walls become of the same height, consider the two walls, having the same height, which are

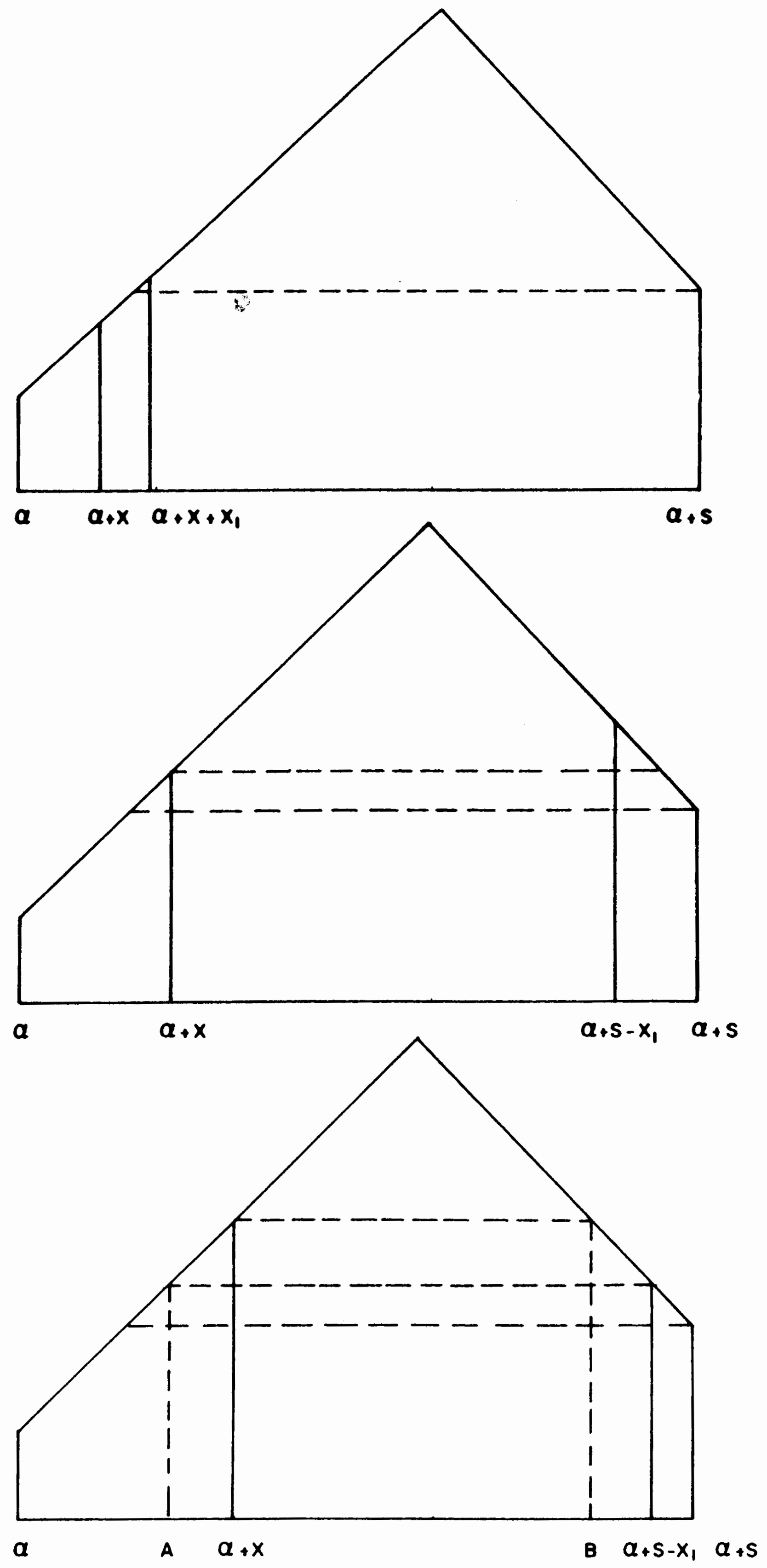

Figure 1. (*Top*) Case 1, (*center*) Case 2, and (*bottom*) Case 3.

furthest away from each other.) This reversal will not affect V at α, and will give an equivalent permutation that will remain greedy when α is increased. Therefore, V is continuous at α.

Next, we prove that $f(\alpha) < f(\alpha + S)$ implies that V is monotone nondecreasing on the interval $(-\infty, \alpha']$. (The proof for the case $f(\alpha) > f(\alpha + S)$ will then follow from the symmetry of f.) Let $\beta < \alpha'$ and, without loss of generality, suppose that $f(\beta) < f(\beta + S)$. The monotonicity is obviously satisfied if $\beta + S < \alpha_2$. Thus, suppose $\beta < \alpha_1 \leq \alpha_2 \leq \beta + S$, and consider the $N + 1$ walls defining $V(\beta)$. Those walls to the right of α_2 or at α_2 can be mapped in a one to one way into the walls to the left of that point in such a way that each wall on the right is mapped into a wall of strictly smaller height. (The shortest wall to the right of or at α_2 is mapped into the shortest wall to the left of α_2, the second shortest wall to the right of or at α_2 is mapped into the second shortest wall to the left of α_2, and so forth.) Because of the concavity and symmetry of f, the sum of the heights of the walls on the right will decrease (as β increases) by an amount that is less than or equal to the sum of the heights of the walls they are mapped into will increase. Since the sum of the unmapped excess walls on the left will obviously not decrease, our local argument, coupled with the continuity of V, completes the proof.

Finally, consider α satisfying $f(\alpha) = f(\alpha + S)$, and suppose that $\bar{\alpha} \neq \alpha$ is a maximum point of V. From the maximality of $\bar{\alpha}$, and the previous discussion, we may assume, without loss of generality, that $f(\bar{\alpha}) = f(\bar{\alpha} + S)$. Since we also have $f(\alpha) = f(\alpha + S)$, the properties of f imply that f is constant on the interval $[\min(\alpha, \bar{\alpha}), \max(\alpha + S, \bar{\alpha} + S)]$. In particular, $V(\alpha) = V(\bar{\alpha})$ and α is also a maximum point of V.

Proof of Theorem A.1. By an inductive argument, it is enough to prove that any permutation π that places a nonminimal room at a lowest end and proceeds thereafter greedily on the remaining subhouse, can be improved by some π' that places a minimal room at a lowest end. For concreteness, suppose that $x_1 \leqq x_2 \leqq \cdots \leqq x_N$, and

$$f(\alpha) \leqq f(\alpha + S). \tag{A.2}$$

Let π be a permutation that first places (at α) a room of length $x > x_1$. We may assume that on the remaining house, with floor $[\alpha + x, \alpha + S]$, the permutation behaves greedily. There are three cases to consider (see Figure 1):

Case 1: $f(\alpha + x) \leqq f(\alpha + S)$ and π places a room of length x_1 in the second place.

Case 2: $f(\alpha + x) \geqq f(\alpha + S)$, π places a room of length $y > x_1$ in the second place and $f(\alpha + x) \leqq f(\alpha + S - x_1)$.

Case 3: $f(\alpha + x) \geqq f(\alpha + S)$, π places a room of length $y > x_1$ in the second place and $f(\alpha + x) > f(\alpha + S - x_1)$.

In Case 1, exchange the first two rooms. Since f is monotone nondecreasing on $[\alpha, \alpha + x]$, $f(\alpha + x_1) \le f(\alpha + x)$. The change in F produced by this exchange is the difference $f(\alpha + x_1) - f(\alpha + x) \le 0$. In Case 2, consider the subhouse with floor $[\alpha + x, \alpha + S - x_1]$. Keeping the lengths of its rooms constant and their inner order greedy, "slide" the subhouse to the left until its leftmost point is at a distance x_1 to the right of α. By Lemma A.1, this sliding process does not increase the value of F. The result is a permutation, as required. Finally, in Case 3, we cannot slide the house as in Case 2 because that may increase F for some sliding distance. To circumvent this difficulty, observe that the subhouse with floor $[\alpha + x, \alpha + S - x_1]$ is the mirror image (with respect to the axis of symmetry of f) of another subhouse, with floor $[A, B]$ (see Figure 1). Note that the induced assignment of the $N - 2$ rooms to the interval $[A, B]$ is greedy and has the same V value as the original assignment of these rooms to the interval $[\alpha + x, \alpha + S - x_1]$. Thus, consider the subhouse with the floor $[A, B]$ instead of that with floor $[\alpha + x, \alpha + S - x_1]$. Now, since $f(A) = f(\alpha + S - x_1) < f(\alpha + x) = f(B)$, the sum of the heights of the walls does not increase when we slide this subhouse to the left (as in Case 2) until its leftmost point is at a distance x_1 to the right of α. (Note that $f(A) = f(\alpha + S - x_1)$ and $f(\alpha) \le f(\alpha + S)$ imply that A is at a distance of at least x_1 from α.) Again, as required, the result is a greedy permutation.

REFERENCES

BAKER, K. R. 1974. *Introduction to Sequencing and Scheduling.* Wiley, New York.

GAREY, M. R. AND D. S. JOHNSON. 1979. *Computers and Intractability: A Guide to NP-Completeness.* Freeman, San Francisco.

KONHEIM, A. G., AND I. MEILIJSON. 1980. Greed Is Good, Research Report RC 8505, Mathematical Sciences Department, IBM Thomas J. Watson Research Center.