SIAM J. COMPUT.
Vol. 8, No. 2, May 1979

0097-5397/79/0802-0012 $01.00/0

NEARLY ON LINE SCHEDULING OF A UNIFORM PROCESSOR SYSTEM WITH RELEASE TIMES*

SARTAJ SAHNI† AND YOOKUN CHO†

Abstract. An $O(m^2n + mn \log n)$ nearly on line algorithm to preemptively schedule n independent tasks on m uniform processors is presented. It is assumed that there is a release time associated with each task. No task may be started before its release time. All tasks must be completed by a common due time (if possible). Our algorithm generates schedules having $O(nm)$ preemptions in the worst case. The algorithm can also be used to minimize maximum lateness even for the case when all jobs have the same release time but different due times.

Key words. independent tasks, uniform processors, preemptive schedule, release time, common due time, complexity

1. Introduction. A *uniform processor system* $P = \{P_1, P_2, \cdots, P_m\}$ is a set of m processors (machines). Associated with each processor, P_i, is a speed s_i, $s_i > 0$, $1 \leqq i \leqq m$. Processor P_i can perform s_i units of processing in one unit of time. When $s_i = s_{i+1}$, $1 \leqq i < m$, P is said to be a system of *identical processors*. Let T be a set of n independent tasks. Let t_i, r_i and d_i respectively be the processing requirement, release time and due time of task i, $1 \leqq i \leqq n$.

A *DD-schedule* for T is an assignment of tasks to processors such that (i) no processor is required to process more than one task at any time, (ii) no task is simultaneously processed on more than one processor, (iii) the processing of no task begins before its release time and (iv) all tasks are completed by their due times. Note that not all task sets have *DD*-schedules on a given processor system.

A nearly on line algorithm to find a *DD*-schedule (if one exists) is an algorithm which, for every distinct release time r_i, determines the schedule from 0 to r_i without knowledge of the jobs released on or after r_i.

Many researchers have studied the problem of obtaining *DD*-schedules (when they exist). Rinnooy Kan [6] shows that the problem of determining the existence of nonpreemptive *DD*-schedules is *NP*-Complete. McNaughton's algorithm [8] can be used to obtain preemptive *DD*-schedules for systems of identical processors when the task set T has only one distinct release time and one distinct due time. Gonzalez and Sahni [3] present an $O(n + m \log m)$ algorithm that works for uniform processor systems when T has only one distinct release time and one distinct due time. For the case when all tasks have the same release time (but may have different due times), Horn [4] presents an $O(n^3)$ algorithm to obtain preemptive *DD*-schedules for identical processors. A faster algorithm ($O(n \log mn)$) for this case may be found in [9]. Under the same assumptions on T, Sahni and Cho [10] obtain an $O(n \log n + mn)$ algorithm for uniform processors. Since, in all the cases cited so far all tasks are released at the same time, all the algorithms obtained are, of necessity, on line.

For the case when no restriction is placed on the task set T, Horn [4] presents an $O(n^3)$ algorithm for preemptive schedules on identical processors. Bruno and Gonzalez [1] present a similar algorithm for a system of two uniform processors. Neither of these two algorithms is on line. In fact, it is known [9] that no nearly on line algorithm exists when tasks are allowed to have arbitrary release and due times.

* Received by the editors October 10, 1977. This work was supported in part by the National Science Foundation under Grant MCS 76-21024.

† Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.