Abstract.

This paper discusses the problem of developing schedules for the execution of computer programs in a batch oriented multiprogramming environment. An approximate approach to the problem is developed by heuristically reducing the dimensionality of the problem to a sequential optimization problem. The superiority of this heuristic criterion over five other commonly used criteria is shown. A numerical example is given.

1. Introduction.

In a multiprogrammed computer installation the computer installations manager has great difficulties in determining what his physical computer needs are, and how much computer power he can squeeze out of his present machine before he needs to acquire additional facilities. This manager would greatly benefit from a tool that enables him to maximize the amount of work the computer can perform in one day. This paper presents a solution to this problem for a batch oriented environment by developing an algorithm that sequences the programs to be run during the day in such a way that the multiprogramming capabilities of the computer are fully utilized.

2. Definition of the Multiprogramming Scheduling Problem.

The scheduling problem can be defined as follows:
A. On a given day, n computer programs are to be executed. Each of these n programs:
a. may or may not have a deadline,
b. requires a fixed amount of contiguous core storage,
c. has a known CPU time,
d. has a known single thread execution time,
e. has known precedence requirements,
f. has known device requirements,
g. must remain in the computer until the end of the job is reached.

B. The computer can process more than one program at a time, with the following qualifications:
   a. device and core restrictions cannot be violated,
   b. delays will occur as program segments seek to access devices already in use,
   c. dynamic reallocation of core is forbidden.

C. The scheduler is to determine the start and finish times for all programs, subject to the above constraints, so that total machine time is minimized.

D. The scheduler should be efficient enough to facilitate re-scheduling whenever additional programs or erroneous execution time estimates make this necessary. This implies low core requirements and high computational speed for an implemented scheduler.

3. The Scheduling Algorithm.

3.1. The approach.

Many of the difficulties encountered in the development of schedules for multiprogramming situations result from the facts that (1), availability of programs change over the scheduling period, and (2), resource availability (tapedrives, core storage, etc.) and execution times are dependent upon the mix of programs in the computer at all times. Given the current state of the art of scheduling and line-balancing theory it is not feasible to develop an exact optimal scheduling algorithm for this situation when the additional requirements of problem size and solution speed are considered. Fortunately it is possible heuristically to reduce the magnitude of this problem to a manageable size while still obtaining close to optimal schedules. This is achieved by introducing a heuristic, dynamically changing, objective function which describes the overall utility of introducing a program at a given time. Programs are chosen to enter core in a sequence determined by the value of this objective function. The terms in this function (discussed in detail in Section 4) are designed to maximize machine utilization as well as to avoid excessive tardiness.