A Research Report submitted to National Science Council

Research Topic: Design and Analysis of Branch-and-Bound Algorithms for some Single Machine Sequencing Problems

Grant Number: NSC 88-2416-H-002-052

Author: Tsung-Chyan Lai

Correspondence: Department of Business Administration, National Taiwan University; email: tclai@ccms.ntu.edu.tw

Key Words: *sequencing/scheduling*, *single machine*, *NP-hard*, *branch-and-bound algorithms*, *lower bounds*, *upper bounds*, *heuristics*, *worst-case analysis*

In this report we outline some of our research results on this research project. We first present some results on a fundamental single machine sequencing problem in which each job j, $j = 1, 2, \ldots, n$, has a release date r_j (arrives at r_j) and a processing time p_j on the machine (occupies an uninterrupted duration of p_j on the machine) and the objective is to minimize the total (average) completion time. Let C_j denote the completion time of job j on the machine (at which job j is finished processing on the machine). The problem is known to be NP-hard (Lenstra et al. 1977) is denoted by $1/r_j/\sum C_j$ (Lawler et al. 1993).

Branch and bound is one of the most effective enumerative algorithms for solving NP-hard scheduling problems up to a practical size. The state-of-the-art branch-and-bound algorithm for solving problem $1/r_j/\sum C_j$ is due to Chu (1992), which can solve the problem up to 100 jobs. In this research we present a branch-and-bound algorithm which can solve problem $1/r_j/\sum C_j$ more than 100 jobs. Some distinguished features in our branch-and-bound algorithm include: (1) some simple heuristics (upper bounding procedures), (2) a new branching (decomposition) scheme, (3) some lower bounding procedures, and more importantly (4) an effectively and efficiently sophisticated upper bounding procedure.

Two well-known simple heuristics are the ECT (earliest completion time) rule and the PS (preemptive schedule) rule. In the ECT rule, at time zero or whenever the machine is idle, a job (among the unsequenced jobs) is selected for processing next by choosing (among the set of unsequenced jobs) the job with the smallest earliest completion time. In the PS rule, jobs

are ordered in increasing order of their completion times in the preemptive schedule, where a preemptive schedule is generated by using the SRPT (shortest remaining processing time) rule. Lai (1998) showed that the ECT rule has a worst-case performance ratio of 2 and the SRPT rule yields a lower bound within $\frac{13}{18}$ of optimal schedule length. Wein et al. (1998) showed that the PS rule has a worst-case performance ratio of 2.

In this research we present some additional simple heuristics. According to our observation, it seems that the ECT rule and the PS rule can complement each other. We thus adopt a compound heuristic consisting of the ECT rule and the PS rule, where we use the ECT rule and the PS rule to separately generate two solutions and then choose the better one as our solution. We conjecture that this compound heuristic has a worst-case performance ratio of $\frac{3}{2}$. A seemingly worst-case example is as follows with $n = 2$: $r_1 = 0$, $p_1 = 1 + \epsilon$, $r_2 = 1$, $p_2 = 0$, where $\epsilon > 0$ is arbitrarily small. Our second simple heuristic is a modified ECT rule, denoted by MECT rule. In the MECT rule, at a decision time point t the set S of unsequenced jobs are classified into two disjoint subsets, denoted by subsets S_1 and S_2, where S_1 consists of the set of jobs that have arrived at t and S_2 consists of the set of jobs that have not arrived yet at t. Let j_1 denote the job with the smallest processing time in S_1, where ties are broken by choosing the job with smallest index, and let j_2 denote the job with the earliest completion time, where ties are broken by choosing the one with the smallest release date. If job j_1 completes earlier than j_2 or job j_1 completes no earlier than twice of its processing time (i.e., $2p_{j_1}$), then job j_1 is choosen for processing next; otherwise, job j_2 is chosen for processing next. We conjecture that the MECT rule has a worst-case performance ratio of $\frac{5}{3}$. A seemingly worst-case example is as follows with $n = 4$: $r_1 = 0$, $p_1 = 1 + \epsilon$, $r_2 = 1$, $p_2 = 0$, $r_3 = 1$, $p_3 = 1 + \epsilon$, $r_4 = 1$, $p_4 = 0$, where $\epsilon > 0$ is arbitrarily small.

Our third simple heuristic is an improved ECT rule, denoted by IECT rule. At a decision time point $t \leq p_{j_1}$, the IECT rule chooses j_1 if job j_1 will complete no later than j_2; otherwise, job j_2 is chosen. At a decision time point t with $p_{j_1} < t < 2p_{j_1}$, the IECT rule chooses job j_1 if the difference between the completion times of j_1 and j_2 is less than or equal the difference between the updated release dates (relative to time t) of j_2 and j_1; otherwise, job j_2 is chosen. At a decision time point $t \geq 2p_{j_1}$, the IECT rule always chooses job j_1. We conjecture that the IECT rule has a worst-case performance ratio of $\frac{3}{2}$. A seemingly worst-case example is as follows with n being arbitrarily large: $r_1 = 0$, $p_1 = 1 + \epsilon$, $r_2 = 1$, $p_2 = 0$, $r_3 = \frac{3}{2}$, $p_3 = 0$, $r_4 = 2$, $p_4 = 0$, $r_j = 2 + \epsilon$, $p_j = 0$, $j = 5, \ldots, n$, where $\epsilon > 0$ is arbitrarily small. Our fourth simple heuristic is the SI (sequential indexing) rule, which can be described as follows. At any decision time point t, let job

j_0 denote the job among the unscheduled jobs with an earliest completion time (ties are broken by choosing the one with a smaller release date and further ties are broken by choosing the one with a smaller index), let A_t denote the set of unscheduled jobs that have arrived before job j_0, let $B_t \subseteq A_t \cup \{j_0\}$ denote the candidate set of jobs, where the candidate set of jobs B_t denotes the set of jobs to be chosen for processing next and for a candidate job $j \in A_t \cup \{j_0\}$ there does not exist another job $i \in A_t \cup \{j_0\}$ that has a smaller updated release date and a smaller processing time. At any decision time point t and for any job $j \in B_t$, let $U_j \subseteq A_t \cup \{j_0\}$ denote the set of jobs that have a smaller release date than job j and let $V_j \subseteq A_t \cup \{j_0\}$ denote the set of jobs that have an earlier completion time than job j. At any decision time point t, let I_j, $j \in B_t$, denote the indexing value of job j, where I_j heuristically represents the cost of choosing job j for processing next with $I_j = \sum_{h \in U_t}(r_j - r_h) + \sum_{h \in V_t}(r_j + p_j - r_h - p_h)$. At any decision time point t, the SI rule chooses among B_t the one with the smallest indexing value I_j.

There are two phases in our new branching scheme. In phase one, a branching decision variable corresponds to whether a job should begin a processing at its release date. In phase two, we adopt the branching scheme as that in Chu (1992). We show that the SRPT rule is also qualified for a lower bounding procedure under the side constraint that some time intervals have been reserved for some jobs. We present several improved SRPT rules. The first one, denoted by ISRPT(1) rule, is to iteratively fix each job j at the time interval $[r_j,\ r_j + p_j]$ and then select the minimum one as the lower bound. The second one, denoted by ISRPT(2) rule, is to improve the lower bound by trying two possible precedence relations of each pair of jobs and then select the best lower bound among all pairs. It seems very difficult to establish the worst-case bounds for the ISRPT(1) and ISRPT(2) rules. Another direction for improving the bound generated by the SRPT rule adopts the idea of charging a cost for each preemption. The issue is how to charge each preemption such that a valid lower bound can still be obtained. In this research we also develop a pricing scheme to charge each preemption and prove its validity in providing a lower bound.

We also present a sophisticated upper bounding procedure which provides a better bound at the expense of more computational time. We partition the n jobs into two sets S_1 and S_2, where each job j in S_1 occupies the interval $[r_j,\ r_j + p_j]$ while each job in S_2 is sequenced according to the rule: at a decision time point t, sequence the job with the smallest processing time among the set of jobs that have arrived. Our upper bounding procedure provides a systematic way of exploring each such partition and then select the partition that generates the best upper bound.

References

Chu, C. 1992. Efficient Heuristics to Minimize Total Flow Time with Release Dates. *Operations Research Letters* **12**, 321-330.

Lai, Tsung-Chyan 1999. Worst-Case Results for Minimizing Total Completion Time on a Single machine with Release Dates, *Naval Research Logistics* (under revision).

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. 1993. "Sequencing and scheduling: algorithms and complexity", in: S.C. Graves et al. (Eds.), *Handbooks in Operations Research and Management Science*, *Vol. 4*, 445-522.

Lenstra, J.K., A.H.G. Rinnooy Kan, and P. Brucker. 1977. Complexity of Machine Scheduling-Problems. *Annals of Discrete Mathematics* **4**, 281-300.

Wein, et al., Mathematical Programming, 1998.