An Inclusion-Exclusion based algorithm for the permutation flowshop scheduling problem

Olivier Ploton, Vincent t'Kindt

An Inclusion-Exclusion based algorithm for the permutation flowshop scheduling problem

Olivier Ploton[1], Vincent T'kindt[1]

Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée
(LIFAT, EA 6300), ERL CNRS 7002 ROOT, Tours, France
{olivier.ploton,vincent.tkindt}@univ-tours.fr

Keywords: flowshop, exponential algorithms, Inclusion-Exclusion.

1 Introduction

In this paper we are interested in minimizing the makespan of a permutation flowshop schedule. Following the notation of Graham et al. (1979), this problem is denoted by $F|prmu|C_{\max}$. In this problem, there are n jobs to be scheduled on m machines. Each job must be processed on machines 1 to m, in this order, and each machine can process only one job at a time. All machines must process jobs in the same order, and a schedule is essentially defined by this order. We note O_{ij}, $i \in \{1 \dots n\}$, $j \in \{1 \dots m\}$, the operation of job i on machine j, which has a non-negative integer processing time p_{ij}. For any given schedule, we define C_{ij} as the completion time of O_{ij} in this schedule. The makespan is the maximum completion time $C_{\max} = \max_{1 \leq i \leq n} C_{im}$. The objective is to find an optimal solution which minimizes the makespan.

We focus on the time and space worst-case complexities of algorithms to solve the $Fm|prmu|C_{\max}$ problem, i.e. when the number of machines is a parameter of the instance. The size of an instance $\mathcal{I}$ is the number of jobs n. The measure of an instance is the sum of its processing times, i.e. $||\mathcal{I}|| = \sum_{i,j} p_{ij}$.

Many algorithms use the branch-and-bound technique, along with specific optimizations. The bounding functions used in these branch-and-bound algorithms are more and more precise as time goes, and some of these algorithms have the best known practical performances (Ladhari and Haouari 2005, Ritt 2016, Gmys et al. 2020). While efficient in practice, they often have a worst-case time complexity bound comparable to the $O^*(n!)$ complexity of the brute-force algorithm.

From a theoretical point of view, few algorithms have been proposed in order to provide better worst-case complexity bounds. Jansen et al. (2013) describe a very general algorithm class, based on a dynamic programming technique on (unordered) sets of jobs. They get time and space worst-case complexities with respect to the number of operations $m \times n$, which translates into $O^*(2^{O(n)}||\mathcal{I}||^{O(1)})$ for each fixed number of machines. Shang et al. (2018) give a more precise result in the particular case of the $F3|prmu|C_{\max}$ problem, by a fine analysis of the number of critical paths in a schedule. They obtain time and space complexities in $O^*(2^n||\mathcal{I}||)$.

Our main contribution in this paper is an algorithm which, for any fixed number of machines, runs with a moderate exponential worst-case time complexity and requires only pseudopolynomial space. More precisely, the time complexity bound is in $O^*(2^n||\mathcal{I}||^m)$ and the space complexity bound is in $O^*(||\mathcal{I}||^m)$.

The algorithms we describe use the Inclusion-Exclusion technique. This rather old combinatorics formula received a pioneer application to computer science by Karp (1982) and Bax (1993). More recently, Inclusion-Exclusion gained in popularity in operational research (Björklund and Husfeldt 2006, Koivisto 2006). Nederlof (2013) showed the interest of this technique to get polynomial or pseudopolynomial space and moderate exponential

time algorithms. Yet, the Inclusion-Exclusion technique is not widely used for scheduling algorithms. To the best of our knowledge, the only scheduling problem solved by an Inclusion-Exclusion based algorithm is the $1|r_i, \tilde{d}_i|$-problem (Karp 1982, Nederlof 2008). Some scheduling algorithms can be reduced to well-known classic problems solved by an Inclusion-Exclusion algorithm, e.g. the $P||C_{\max}$ problem reduces to the bin-packing problem (Karp 1982) and the $F|nowait|C_{\max}$ problem reduces to the Asymmetric Traveling Salesman Problem (Bax 1993, Karp 1982).

2 The permutation flowshop decision problem

As the makespan is a regular objective, we can restrict to semi-active schedules, where no operation could be scheduled earlier without changing the job order. So, a schedule S is uniquely represented by the sequence of its jobs: $S = (i_1 \dots i_n)$.

Figure 1 presents an annotated Gantt chart showing a solution of the permutation flowshop problem (one line per machine, one color per job).

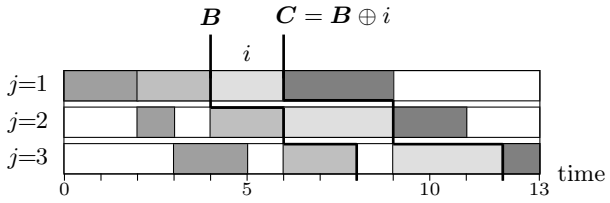

Fig. 1. A permutation schedule

Before executing job i, machines must be done with previous jobs. So, it is useful to consider a more precise version of the problem: machines have release times $(B_j)_{j=1\dots m}$ before which they are busy. Together, they form a time front $\boldsymbol{B} = (B_j)$. We denote by $\boldsymbol{C} = \boldsymbol{B} \oplus i$ the completion times (C_j) of job i on machine j when executed with release times (B_j). These are also the release times of the next job to be processed. With the convention that $C_0 = 0$, we have:

$$C_j = \max(C_{j-1}, B_j) + p_{ij}, \quad \forall\, j = 1 \dots m. \tag{1}$$

Let I be a set of jobs, $\boldsymbol{B}$ a release time front and ε a given threshold on the makespan value. We now want to determine whether or not there exists a schedule using jobs of I, whose makespan is at most ε when run with release times $\boldsymbol{B}$. For that, we shall count the number $N(I, \boldsymbol{B}, \varepsilon)$ of such schedules. We are about to do it using Inclusion-Exclusion.

3 The Inclusion-Exclusion principle

To apply the Inclusion-Exclusion principle, following Fomin and Kratsch (2010), consider a problem of size n in which a solution is represented by a permutation. Relax this problem by allowing any list of length n, with possible duplicates or missing elements. For any $J \subset I$, count valid lists using only elements of J. By Inclusion-Exclusion, we deduce the number of solutions of the initial problem.

We apply this principle to count schedules viewed as lists of jobs. We denote by $\mathfrak{S}_n$ the set of permutations schedules, where all jobs appear once, and I^n is the set of relaxed

schedules, where there may be duplicate or missing jobs. Define $E \subset I^n$ as the set of relaxed schedules whose makespan is at most ε when run with release times $\boldsymbol{B}$. We derive:

$$\underbrace{\operatorname{card} E \cap \mathfrak{S}_n}_{N(I,\boldsymbol{B},\varepsilon)} = \sum_{J \subset I} (-1)^{|I|-|J|} \underbrace{\operatorname{card} E \cap J^n}_{N_J(\boldsymbol{B},\varepsilon)} \tag{2}$$

We determine each term $N_J(\boldsymbol{B}, \varepsilon)$ by dynamic programming. We compute $N_{J,\varepsilon}[\ell, \boldsymbol{B}]$ as the number of relaxed schedules of length ℓ, using only jobs of J, whose makespans are at most ε when run with release times $\boldsymbol{B}$. We have:

$$N_J(\boldsymbol{B}, \varepsilon) = N_{J,\varepsilon}[n, \boldsymbol{B}] \tag{3}$$

$$N_{J,\varepsilon}[\ell, \boldsymbol{B}] = \sum_{i \in J} N_{J,\varepsilon}[\ell{-}1, \boldsymbol{B} \oplus i] \text{ if } B_m \leq \varepsilon, \text{ 0 otherwise} \qquad \forall\, \ell = 1 \ldots n \tag{4}$$

$$N_{J,\varepsilon}[0, \boldsymbol{B}] = 1 \text{ if } B_m \leq \varepsilon, \text{ 0 otherwise.} \tag{5}$$

4 From a feasible makespan value to an explicit solution

The optimal makespan can be deduced from the decision problem. It is:

$$C_{\max}^{\text{opt}} = \min\{\varepsilon \mid N(I, (B_j{=}0), \varepsilon) > 0\} \tag{6}$$

It can be computed by using a dichotomic search.

Once the optimal makespan is known, we can determine an explicit solution step by step. We write (Algorithm 1) the recursive function $\mathit{Solution}(\sigma, \boldsymbol{B}, I)$, where σ is the sequence of already scheduled jobs, $\boldsymbol{B} = (B_j)_j$ are the completion times of already scheduled jobs, and I is the set of jobs to be scheduled after $\boldsymbol{B}$. We use the decision algorithm as an oracle to systematically choose a job outside σ leading to a feasible solution. We denote by () the empty sequence and by $\cdot$ the concatenation of sequences. We obtain an optimal solution of the $Fm|prmu|C_{\max}$ problem by calling $\mathit{Solution}(\sigma{=}(), (B_j{=}0), I{=}\{1 \ldots n\})$.

```
Function Solution(σ, B, I):
    if I ≠ ∅ then
        for i ∈ I do
            if N(I \ {i}, B ⊕ i, C_max^opt) > 0 then
                return Solution(σ · i, B ⊕ i, I \ {i})
    else
        return σ
```

Algorithm 1: Computation of a feasible schedule

5 Worst-case complexities

We now evaluate the worst-case time and space complexities of our algorithms:

- Each component of the timefront $\boldsymbol{B}$ involved in the dynamic programming equations (3), (4), (5) is bounded by the sum of the processing times, i.e. $||\mathcal{I}||$. So, the number of involved states is in $O^*(||\mathcal{I}||^m)$.
- The decision problem uses 2^n independent dynamic programming computations and it can be solved in $O^*(2^n||\mathcal{I}||^m)$ time $O^*(||\mathcal{I}||^m)$ space.
- Dichotomic computation of $C_{\max}^{\text{opt}}$ is in $O^*(2^n||\mathcal{I}||^m \log ||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||^m)$ space.
- When $C_{\max}^{\text{opt}}$ is known, computation of an optimal solution of the $Fm|prmu|C_{\max}$ problem is in $O^*(2^n||\mathcal{I}||^m)$ time and $O^*(||\mathcal{I}||^m)$ space.
- The global algorithm is in $O^*(2^n||\mathcal{I}||^m \log ||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||^m)$ space.

6 Conclusions

In this paper, we study exact algorithms to minimize the makespan of permutation flowshop schedules, and we focus on bounding worst-case time and space complexities. These complexities are evaluated for a fixed number of machines m using job number n as the instance size and the sum of the processing times as an instance measure $||\mathcal{I}||$. The best general time and space complexity bounds proved so far is due to Jansen et al. (2013). It is $O^*(2^{O(n)}||\mathcal{I}||^{O(1)})$, for each fixed m. Shang et al. (2018) proved a more precise bound of $O^*(2^n||\mathcal{I}||)$, for the particular case of 3 machines.

We describe an Inclusion-Exclusion based algorithm for the $Fm|prmu|C_{\max}$ problem, using dynamic programming for enumerations. We prove that, for every fixed m, its worst-case space complexity is pseudopolynomial, with bound $O^*(||\mathcal{I}||^m)$, and its worst-case time complexity is moderately exponential, with bound $O^*(2^n||\mathcal{I}||^m)$.

From this piece of research, several future research directions can be outlined: how to optimize the computation of the sum involved in the Inclusion-Exclusion principle ? How to get tighter bounds on the number of states used in the dynamic programming algorithm ? These questions are of great importance to derive better worst-case complexity bounds.

References

Bax E.T., 1993, "Inclusion and exclusion algorithm for the Hamiltonian path problem", *Information Processing Letters*, Vol 17(4), pp 203–207.

Björklund A., T. Husfeldt, 2006, "Inclusion-exclusion algorithms for counting set partitions", *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pp 575–582.

Fomin F.V., D. Kratsch, 2010, "Exact exponential algorithms", Springer.

Gmys J., M. Mezmaz, N. Melab, D. Tuyttens, 2020, "A computationally efficient Branch-and-Bound algorithm for the permutation flow-shop scheduling problem", *European Journal of Operational Research*, Vol 284(3), pp 814–833.

Graham R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, 1979, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications*, Vol 5, pp 287–326.

Jansen K., F. Land, K. Land, 2013, "Bounding the Running Time of Algorithms for Scheduling and Packing Problems", *Algorithms and Data Structures - 13th International Symposium*, pp 439–450.

Karp R.M., 1982, "Dynamic Processing meets the principle of inclusion and exclusion", *Operational Research Letters*, Vol 1(2), pp 49–51.

Koivisto M., 2006, "An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion", *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pp 583–590.

Ladhari T., M. Haouari, 2005, "A computational study of the permutation flow shop problem based on a tight lower bound", *Computers & Operations Research*, Vol 32, pp 1831–1847.

Nederlof J., 2008, "Inclusion-exclusion for hard problems", *Master Thesis*, Utrecht University.

Nederlof J., 2013, "Fast Polynomial-Space Algorithms Using Inclusion-Exclusion", *Algorithmica*, Vol 65, pp 868–884.

Ritt M., 2016, "A branch-and-bound algorithm with cyclic best-first search for the permutation flow shop scheduling problem", *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp 872–877.

Ryser H.J., 1963, "Combinatorial mathematics", *The Carus Mathematical Monographs, No 14*, The Mathematical Association of America.

Shang L., C. Lenté, M. Liedloff, V. T'kindt, 2018, "Exact exponential algorithms for 3-machine flowshop scheduling problems", *Journal of Scheduling*, Vol 21, pp 227–233.