Moderate Exponential-time Quantum Dynamic Programming Across the Subsets for Scheduling Problems

Camille Grange[1,2], Michael Poss[1], Eric Bourreau[1], Vincent T'kindt[3], Olivier Ploton[3]

[1] University of Montpellier, LIRMM, CNRS, France
{camille.grange, michael.poss, eric.bourreau}@lirmm.fr
[2] SNCF, Technology, Innovation and Group Projects Department, France
[3] University of Tours, LIFAT, France
{tkindt, olivier.ploton}@univ-tours.fr

Keywords : *quantum optimization, Grover algorithm, dynamic programming, scheduling.*

1 Introduction

The interest in quantum computing to solve combinatorial optimization problems has been growing for several years in the operational research community. More precisely, we distinguish two branches. The first one relates to heuristics, which can be implemented on current noisy quantum computers because the quantum part can be made rather small. The second branch relates to *exact* algorithms. Unlike the previous algorithms, it is impossible to implement them today but theoretical speed-ups have been proved for several types of problems.

The most emblematic algorithm of the latter branch is Grover Search [4], which achieves a quadratic speed-up when searching for a specific element in an unsorted table. Durr and Hoyer [3] use Grover Search as a subroutine for a hybrid algorithm, Quantum Minimum Finding (QMF), that finds the minimum of an unsorted table. Later, Ambainis et al. [1] combine QMF with dynamic programming to address NP-hard vertex ordering problems such as the Traveling Salesman Problem. These problems satisfy a specific property which implies that they can be solved by classical dynamic programming in $\mathcal{O}^*(c^n)$, where $\mathcal{O}^*$ is the usual asymptotic notation that ignores the polynomial factors, and c is usually not smaller than 2. The hybrid algorithm reduces the complexity to $\mathcal{O}^*(c^n_{\text{quant}})$ for $c_{\text{quant}} < c$. Following this work, other NP-hard problems have been tackled with the idea of combining QMF and classical dynamic programming, such as for the Steiner Tree problem (Miyamoto et al. [6]).

The purpose of this work is to provide a general method, a quantum-classical algorithm, adapting the seminal idea of Ambainis et al. [1], to reduce the worst-case time complexity of solving problems on which the Dynamic Programming Across the Subsets (DPAS) can be applied. These types of problems are directly inspired by NP-hard scheduling problems described by T'kindt et al. [9] but the mathematical formulations throughout this work aim to be as generic as possible, leading the proposed algorithm to be applicable to a broader class of problems. In particular, it applies to problems with temporal constraints and non-linear objective functions found in the scheduling literature. Herein, we focus on two single-machine scheduling problems ($1|\tilde{d}_j|\sum_j w_j C_j$ and $1|r_j|\sum_j w_j U_j$, defined later) and show that our algorithm reduces the worst-case time complexity of problems compared to the current best-known classical algorithms, sometimes at the cost of an additional pseudo-polynomial factor. Due to space limit, all proofs of propositions are omitted. We only provide a sketch of our main result.

2 Dynamic Programming Across the Subsets (DPAS)

Let us consider a scheduling problem of n jobs

$$\mathcal{P}: \qquad \min_{\pi \in \Pi} f(\pi)\,,$$

where $\Pi \subseteq S_{[n]}$ is the set of feasible permutations of $[n] := \{1, \ldots, n\}$ according to given constraints and f is the objective function. We consider a related problem P useful for deriving the dynamic programming recursion, for which we specify the instance: for $J \subseteq [n]$ and $t \in \mathbb{Z}$, we define

$$P(J,t): \qquad \min_{\pi \in \Pi(J,t)} f(\pi, J, t) \tag{1}$$

as the nominal scheduling problem $\mathcal{P}$ that schedules only jobs in J and starts the schedule at time t. Let us note $\mathrm{OPT}[J,t]$ the optimal value of $P(J,t)$. It results that solving $\mathcal{P}$ amounts to solving $P([n],0)$, because $\mathcal{P}$ consists in scheduling all the jobs and starting at time $t=0$.

In this section, we describe two classes of scheduling problems that can be solved by our hybrid algorithm we call Quantum Dichotomic DPAS (Q-DDPAS). They differ in the type of recurrence they satisfy to compute the optimal value $\mathrm{OPT}[J,t]$ for a given $J \subseteq [n]$ and $t \in \mathbb{Z}$. The first class is composed of problems for which the constraints are compatible with the *addition* of optimal values $\mathrm{OPT}[X,t']$, for $X \subseteq J$ and $t' \in \mathbb{Z}$, whereas the second class contains problems for which the constraints impose the *composition* of the $\mathrm{OPT}[X,t']$.

2.1 Additive DPAS

Let us start by defining problems for which the constraints are compatible with the addition of optimal values of problems on sub-instances, formally defined below. For instance, one can think of deadline or precedence constraints. Problem $\mathcal{P}$, i.e. $P([n],0)$, can be solved by Q-DDPAS if the related problem P satisfies the two recurrences (Add-DPAS) and (Add-D-DPAS) below. Henceforth, we denote by $2^{[n]}$ the set of all subsets of $[n]$, and by $[\![a,b]\!]$ the set $\{a,\ldots,b\}$. Let us introduce the first recurrence.

Property 1 (Additive DPAS)**.** *There exists a function $g : 2^{[n]} \times [n] \times T \to \mathbb{R}$, computable in polynomial time, such that, for all $J \subseteq [n]$ and for all $t_0 \in T$,*

$$\mathrm{OPT}[J,t_0] = \min_{j \in J} \Big\{ \mathrm{OPT}[J \setminus \{j\}, t_0] + g(J,j,t_0) \Big\} \tag{Add-DPAS}$$

initialized by $\mathrm{OPT}[\emptyset, t_0] = 0$.

Proposition 2. *Dynamic programming* (Add-DPAS) *solves $\mathcal{P}$ in $\mathcal{O}^*(2^n)$.*

Throughout, we commit a slight abuse of language by letting (Add-DPAS) both refer to the property satisfied by a given optimization problem and to the resulting dynamic programming algorithm.

Property 1 expresses that finding the optimal value of P for jobs in J and starting at time t is done by finding over all jobs $j \in J$ the permutation that ends by j with the best cost value. Function g represents the cost of j being the last job of the permutation. Notice that isolating the last job of the permutation is a usual technique in scheduling. In the second recurrence below, we provide a similar scheme, where instead of one job, we *isolate* half of the jobs in J, turning the computation of g to the resolution of another problem on a sub-instance with $|J|/2$ jobs.

Property 3 (Additive Dichotomic DPAS)**.** *There exist two functions $t_{shift} : 2^{[n]} \times 2^{[n]} \times T \to T$ and $h : 2^{[n]} \times 2^{[n]} \times T \to \mathbb{R}$, computable in polynomial time, such that, for all $J \subseteq [n]$ of even cardinality, and for all $t \in T$,*

$$\mathrm{OPT}[J,t] = \min_{\substack{X \subseteq J \\ |X| = \lceil J/2 \rceil}} \Big\{ \mathrm{OPT}[X,t] + h(J,X,t) + \mathrm{OPT}[J \setminus X, t_{shift}(J,X,t)] \Big\} \tag{Add-D-DPAS}$$

initialized by the values $\mathrm{OPT}[\{j\},t]$ *for each $j \in [n]$ and $t \in T$.*

For a given $X \subseteq J$, the above computes the best permutation of jobs in X starting at time t, and the best permutation of jobs in $J \setminus X$ starting at time t_{shift}, adding the function h that represents the cost of the concatenation between these two permutations.

We use the notation $f_1(n) = \omega(f_2(n))$ if f_1 dominates asymptotically f_2.

Proposition 4. *Dynamic programming* (Add-D-DPAS) *solves* $\mathcal{P}$ *in* $\omega(|T| \cdot 2^n)$.

Notice that if n is not a power of 2, we can still add fake jobs without changing the following conclusion: solving $\mathcal{P}$ with (Add-DPAS) is faster than with (Add-D-DPAS). However, in the next section, we describe a hybrid algorithm Q-DDPAS that improves the complexity of solving $\mathcal{P}$ by combining recurrences (Add-DPAS) and (Add-D-DPAS) with a quantum subroutine.

Before introducing this algorithm, we illustrate the two recurrences above with the NP-hard single-machine scheduling problem with deadline constraints and minimization of the total weighted completion time $(1|\tilde{d}_j|\sum_j w_j C_j)$. For each job $j \in [n]$, we are given a weight w_j, a processing time p_j, and a deadline $\tilde{d}_j$. Let $T = [\![0, p([n])]\!]$, where we note $p(J) = \sum_{i \in J} p_i$. We consider the related problem (1) as follows: for $J \subseteq [n]$ and $t \in T$,

$$\Pi(J,t) = \{\pi \in S_J \,|\, C_j(\pi) \leq \tilde{d}_j - t \,, \forall j \in J\} \,,$$

where C_j is the completion time of job j, and for $\pi \in \Pi(J,t)$:

$$f(\pi, J, t) = \sum_{j \in J} w_j (C_j(\pi) + t) \,.$$

Thus, the optimal solution of $P(J,t)$ is the best feasible solution for jobs in J supposing that the machine is available at time t (instead of 0, as usual). This leads to considering the *effective* deadlines $(\tilde{d}_j - t)$ instead of $\tilde{d}_j$ in the constraints and adding the term $t \cdot \sum_{j \in J} w_j$ in the cost function. Our problem of interest $\mathcal{P}$ is $P([n], 0)$, and one readily verifies that P satisfies both recurrences. Indeed, (Add-DPAS) is valid with: $\forall J \subseteq [n], \forall j \in J, \forall t \in T$,

$$g(J,j,t) = \begin{cases} w_j(p(J \cup \{j\}) + t) & \text{if } \tilde{d}_j \geq p(J \cup \{j\}) + t \\ +\infty & \text{otherwise} \end{cases}$$

where the computation of g is polynomial (linear). Recurrence (Add-D-DPAS) is also valid with the following functions: $\forall X \subseteq J \subseteq [n] : |X| = |J|/2, \forall t \in T$,

$$t_{\text{shift}}(J,X,t) = t + p(X) \quad \text{and} \quad h(J,X,t) = 0 \,,$$

initialized by, for $j \in [n]$ and $t \in T$, $\text{OPT}[\{j\}, t] = \begin{cases} w_j(p_j + t) & \text{if } \tilde{d}_j \geq p_j + t \\ +\infty & \text{otherwise} \end{cases}$

2.2 Composed DPAS

We study now problems whose constraints enable only the *composition* of problems of sub-instances. It mainly concerns scheduling with release date constraints.

To ease the understanding, we begin with a practical example, where $\mathcal{P}$ is the problem of minimizing the total weighted sum of late jobs with release date constraints $(1|r_j|\sum w_j U_j)$. Each job $j \in [n]$ has a weight w_j, a processing time p_j, a due date d_j, and a release date r_j. We define a new related problem P' such that, for a given $\epsilon \in E := \left[\!\!\left[0, \sum_{j=1}^{n} w_j\right]\!\!\right]$, the optimal value $\text{OPT}[J,t,\epsilon]$ of $P'(J,t,\epsilon)$, for $J \subseteq [n]$ and $t \in T := \left[\!\!\left[0, \sum_{j=1}^{n} p_j\right]\!\!\right] \cup \{+\infty\}$, is the minimum makespan for jobs in J beginning at time t where the weighted sum of late jobs is ϵ. More precisely,

$$P'(J,t,\epsilon) : \quad \min_{\pi \in \Pi'(J,t,\epsilon)} C_{\max}(\pi) \,,$$

where $C_{\max}$ is the makespan (maximum completion time), and

$$\Pi'(J,t,\epsilon) = \left\{ \pi \in S_J : C_j(\pi) \geq \max(t, r_j) + p_j \text{ and } \sum_{j \in J} w_j U_j(\pi) = \epsilon \right\},$$

where U_j indicates if job j is late, namely, $U_j = \mathbb{1}_{\{j \text{ is late}\}}$. Thus, our problem of interest is

$$\mathcal{P} = \min_{\epsilon \in E} \{\epsilon : \mathrm{OPT}[[n], 0, \epsilon] < +\infty\}.$$

One can show that P' satisfies the following recurrence, inspired by the work of Lawler [5] for the problem of minimizing the total weighted number of late jobs on a single machine under preemption and release date constraints ($1|r_j, pmtn| \sum w_j U_j$). For $J \subseteq [n]$, $t \in T$, $\epsilon \in E$,

$$\mathrm{OPT}[J,t,\epsilon] = \min_{j \in J} \Big\{ \underbrace{\mathrm{OPT}\Big[\{j\}, \mathrm{OPT}[J \setminus \{j\}, t, \epsilon], 0\Big]}_{\text{job } j \text{ is not late}}, \underbrace{\mathrm{OPT}\Big[\{j\}, \mathrm{OPT}[J \setminus \{j\}, t, \epsilon - w_j], w_j\Big]}_{\text{job } j \text{ is late}} \Big\}.$$

This leads to the more general recurrence below. Henceforth, we consider a generic problem $\mathcal{P}$ related to P' for sets $T, E \subseteq \mathbb{Z}$.

Property 5 (Composed DPAS)**.** *For all $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[J,t,\epsilon] = \min_{\substack{\epsilon' \in E \\ j \in J}} \Big\{ \mathrm{OPT}\Big[\{j\}, \mathrm{OPT}[J \setminus \{j\}, t, \epsilon - \epsilon'], \epsilon'\Big] \Big\}, \qquad \text{(Comp-DPAS)}$$

initialized by the values of $\mathrm{OPT}[\{j\}, t, \epsilon]$ *for all $j \in [n]$, $\epsilon \in E$ and $t \in T$. Notice that for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$, we adopt the convention* $\mathrm{OPT}[J,t,\epsilon] = +\infty$ *for $\epsilon \notin E$.*

Proposition 6. *For $\epsilon_0 \in E$,* (Comp-DPAS) *solves $P'([n], 0, \epsilon_0)$ in $\mathcal{O}^*(|E|^2 \cdot |T| \cdot 2^n)$.*

It directly results from the previous proposition that (Comp-DPAS) solves $\mathcal{P}$ in $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 2^n)$. As for Additive DPAS, we provide a *dichotomic* counterpart of (Comp-DPAS) as follows.

Property 7 (Composed Dichotomic DPAS)**.** *For all $J \subseteq [n]$ of even cardinality, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[J,t,\epsilon] = \min_{\substack{\epsilon' \in E \\ X \in J : |X| = |J|/2}} \Big\{ \mathrm{OPT}\Big[X, \mathrm{OPT}[J \setminus X, t, \epsilon - \epsilon'], \epsilon'\Big] \Big\}, \qquad \text{(Comp-D-DPAS)}$$

initialized by the values of $\mathrm{OPT}[\{j\}, t, \epsilon]$ *for all $j \in [n]$, $t \in T$ and $\epsilon \in E$.*

Proposition 8. *For $\epsilon_0 \in E$,* (Comp-D-DPAS) *solves $P'([n], 0, \epsilon_0)$ in $\omega(|E|^2 \cdot |T| \cdot 2^n)$.*

The previous proposition naturally implies that (Comp-D-DPAS) solves $\mathcal{P}$ in $\omega(|E|^3 \cdot |T| \cdot 2^n)$. As for Additive DPAS, we notice that, with a classical dynamic programming algorithm, the time complexity to solve $\mathcal{P}$ with (Comp-DPAS) is better than with (Comp-D-DPAS). We show next that our hybrid algorithm Q-DDPAS improves these complexities.

3 Hybrid Algorithm Q-DDPAS

In this section, we describe our hybrid algorithm Q-DDPAS adapted from the work of Ambainis et al. [1]. Notice that we design it in the gate-based quantum computing model and we assume to have random access to quantum memory (QRAM). We underline that this latter assumption is strong because QRAM is not yet available on current universal quantum hardware. We begin with the description of Q-DDPAS for problems $\mathcal{P}$ whose related problem P satisfies recurrences (Add-DPAS) and (Add-D-DPAS). Q-DDPAS for problems whose related problem P' satisfies recurrences (Comp-DPAS) and (Comp-D-DPAS) derives directly.

First, let us introduce the Quantum Minimum Finding (QMF) of Durr and Hoyer [3], which constitutes a fundamental subroutine in our algorithm. This algorithm essentially applies several times Grover Search [4].

Definition 9 (Quantum Minimum Finding [3]). *Let $f : [n] \to \mathbb{Z}$ be a function. QMF computes the minimum value of f and the corresponding minimizer* $\arg\min_{i\in[n]}\{f(i)\}$. *The complexity of QMF is $\mathcal{O}\left(\sqrt{n}\cdot C_f(n)\right)$, where $\mathcal{O}(C_f(n))$ is the complexity of computing a value of f.*

Without loss of generality, we assume that 4 divides n. This can be achieved by adding at most three fake jobs and, therefore, does not change the algorithm complexity. Q-DDPAS consists of two steps. First, we compute classically by (Add-DPAS) the optimal values of P on sub-instances of $n/4$ jobs and for all starting times $t \in T$. Second, we call recursively two times QMF with (Add-D-DPAS) to find optimal values of P on sub-instances of $n/2$ jobs starting at any time $t \in T$, and eventually of n jobs starting at $t = 0$ (corresponding to the optimal value of the nominal problem $\mathcal{P}$). Specifically, we describe Q-DDPAS in Algorithm 1.

Algorithm 1: Q-DDPAS for Additive DPAS

Input: Problem P satisfying (Add-DPAS) and (Add-D-DPAS)
Output: OPT$[[n], 0]$ with high probability
begin classical part
  for $X \subseteq [n]$ *such that* $|X| = n/4$, *and* $t \in T$ **do**
1 Compute OPT$[X, t]$ with (Add-DPAS) and store the results in the QRAM;
begin quantum part
2 Apply QMF with (Add-D-DPAS) to find OPT$[[n], 0]$;
3 To get values for the QMF above (the values OPT$[J, t]$ for $J \subseteq [n]$ of size $n/2$ and $t \in T$), apply QMF with (Add-D-DPAS);
4 To get values for the QMF above (the values OPT$[X, t']$ for $X \subseteq [n]$ of size $n/4$ and $t' \in T$), get them on the QRAM

Theorem 10. *Q-DDPAS for Additive DPAS solves $\mathcal{P}$ in $\mathcal{O}^*(|T| \cdot 1.754^n)$.*

Proof. We provide next a sketch of the proof.

- Classical part: computing all OPT$[X, t]$ for all X of size $n/4$ and for all $t \in T$ (Step 1) is done by (Add-D-DPAS) in time $\mathcal{O}^*\left(|T| \cdot \sum_{k=1}^{n/4} k\binom{n}{k}\right) = \mathcal{O}^*(|T| \cdot 2^{0.811n})$.
- Quantum part: according to QMF complexity (Definition 9), computing OPT$[[n], 0]$ with QMF (Step 2) is done in $\mathcal{O}\left(\sqrt{\binom{n}{n/2}} \cdot C_1(n)\right)$, where $C_1(n)$ is the complexity of computing OPT$[J, t]$ for a J of size $n/2$ and $t \in T$. *The essence of the quantum advantage here is that we do not need to enumerate all sets J and all time t but we apply the QMF in parallel to all at once.* Notice that $\binom{n}{n/2}$ is the number of balanced bi-partitions of $[n]$, namely the number of elements in (Add-D-DPAS) when computing OPT$[[n], 0]$. Thus, $C_1(n)$ is exactly the complexity of QMF applied on Step 3, namely $C_1(n) = \mathcal{O}\left(\sqrt{\binom{n/2}{n/4}} \cdot C_2(n)\right)$ where $C_2(n)$ is the complexity of computing OPT$[X, t']$ for X of size $n/4$ and $t' \in T$. Those values are already computed and stored in the QRAM (Step 4), namely, $C_2(n) = \mathcal{O}^*(1)$. Thus, the quantum part complexity is $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}\right) = \mathcal{O}^*(2^{0.75n})$.

Eventually, Q-DDPAS complexity is the maximum of the classical and the quantum part complexity. Specifically, the total complexity is $\mathcal{O}^*(|T| \cdot 2^{0.811n}) = \mathcal{O}^*(|T| \cdot 1.754^n)$. □

The resolution of a problem $\mathcal{P}$ satisfying (Comp-DPAS) and (Comp-D-DPAS) derives naturally. It amounts to iterate over all $\epsilon \in E$ and solves each $P([n], 0, \epsilon)$ with Q-DDPAS relying on the Composed DPAS recurrences in this case, namely replacing (Add-DPAS), respectively (Add-D-DPAS), by (Comp-DPAS), respectively (Comp-D-DPAS).

Theorem 11. *Q-DDPAS for Composed DPAS solves $\mathcal{P}$ in $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 1.754^n)$.*

We summarize in the first part of Table 1 the complexities of solving the studied scheduling problems with Q-DDPAS and compare them with the complexities of the best-known classical algorithms. Q-DDPAS improves the exponential part of the complexity, sometimes at the cost of a pseudo-polynomial factor. We complete the second part of Table 1 with the application of Q-DDPAS to other single-machine scheduling problems ($1||\sum w_jT_j$, $1|prec|\sum w_jC_j$ for Additive DPAS, and $1|r_j|\sum w_jC_j$ for Composed DPAS). The last result is an adaptation of Q-DDPAS to decision problems, applied to the 3-machine flowshop ($F3||C_{\max}$). All these results are detailed in the long version of this work (https://hal.science/hal-04296238v1).

Problem	Q-DDPAS	Best-known classical algorithm
$1\|\tilde{d}_j\|\sum w_jC_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.754^n\right)$	$\mathcal{O}^*(2^n)$ [9]
$1\|r_j\|\sum w_jU_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.754^n\right)$	$\mathcal{O}^*(\sum w_j \cdot \sum p_j \cdot 2^n)$ [7]
$1\|\|\sum w_jT_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.754^n\right)$	$\mathcal{O}^*(2^n)$ [9]
$1\|prec\|\sum w_jC_j$	$\mathcal{O}^*\left(1.754^n\right)$	$\mathcal{O}^*((2-\epsilon)^n)$, for small ϵ [2]
$1\|r_j\|\sum w_jC_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot (\sum p_j)^4 \cdot 1.754^n\right)$	$\mathcal{O}^*(\sum w_j \cdot (\sum p_j)^2 \cdot 2^n)$ [7]
$F3\|\|C_{\max}$	$\mathcal{O}^*\left((\sum p_{ij})^4 \cdot 1.754^n\right)$	$\mathcal{O}^*(3^n)$ [8]

TAB. 1: Comparison of complexities between our hybrid algorithms and the best-known classical algorithms.

References

[1] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *SODA*. SIAM, 2019.

[2] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Scheduling partially ordered jobs faster than 2^n. *Algorithmica*, 68:692–714, 2014.

[3] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[4] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[5] Eugene L Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26:125–133, 1990.

[6] Masayuki Miyamoto, Masakazu Iwamura, Koichi Kise, and François Le Gall. Quantum speedup for the minimum steiner tree problem. In *COCOON*. Springer, 2020.

[7] Olivier Ploton and Vincent T'kindt. Exponential-time algorithms for parallel machine scheduling problems. *Journal of Combinatorial Optimization*, 44(5):3405–3418, 2022.

[8] Lei Shang, Christophe Lenté, Mathieu Liedloff, and Vincent T'Kindt. Exact exponential algorithms for 3-machine flowshop scheduling problems. *Journal of Scheduling*, 21:227–233, 2018.

[9] Vincent T'kindt, Federico Della Croce, and Mathieu Liedloff. Moderate exponential-time algorithms for scheduling problems. *4OR*, pages 1–34, 2022.