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

Camille Grange, Michael Poss, Eric Bourreau, Vincent t'Kindt, Olivier Ploton

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], and Olivier Ploton[3]

[1]University of Montpellier, LIRMM, CNRS, France

[2]SNCF, Technology, Innovation and Group Projects Department, France

[3]University of Tours, LIFAT, France

Abstract

Grover Search is currently one of the main quantum algorithms leading to hybrid quantum-classical methods that reduce the worst-case time complexity for several combinatorial optimization problems. Specifically, the combination of Quantum Minimum Finding (obtained from Grover Search) with dynamic programming has proved particularly efficient in improving the complexity of NP-hard problems currently solved by classical dynamic programming. For these problems, the classical dynamic programming complexity (ignoring the polynomial factors) in $\mathcal{O}^*(c^n)$ can be reduced by a hybrid algorithm to $\mathcal{O}^*(c_{quant}^n)$, with $c_{quant} < c$. In this paper, we provide a bounded-error hybrid algorithm that achieves such an improvement on NP-hard minimization problems for which we give a generic description. We illustrate our approach on a variety of scheduling problems. Moreover, we extend this algorithm to decision problems to tackle the 3-machine flowshop problem. Our algorithm reduces the exponential-part complexity compared to the best-known classical algorithm, sometimes at the cost of an additional pseudo-polynomial factor.

keywords: Quantum optimization, Grover algorithm, Dynamic Programming, Scheduling

*This is an extension of the conference paper of Grange et al. [10].

1 Introduction

The fields of quantum computing and combinatorial optimization are becoming every day more closely linked, thanks to the work of the operations research community that has been focusing on the new quantum paradigm. More precisely, there are two types of quantum algorithms for solving optimization problems. The first type encompasses heuristics, often designed today as hybrid quantum-classical algorithms, such as the class of Variational Quantum Algorithms described by Cerezo et al. [5] or by Grange et al. [11] and, within it, the famous Quantum Approximate Optimization Algorithm (QAOA) of Farhi et al. [8]. Essentially, these algorithms require the optimization problem to be formulated as a QUBO (Quadratic Unconstrained Binary Optimization) and can be implemented on current noisy quantum computers because the quantum part can be made rather small. Among others, the problems of MAX-CUT (Farhi et al. [8]), Travelling Salesman Problem (Ruan et al. [27]), MAX-3-SAT (Nannicini [20]), Graph Coloring (Tabi et al. [31]) and Job Shop Scheduling (Kurowki et al. [16]) are reformulated as QUBO and solved with hybrid heuristics on small instances. However, due to the small size of instances processed today and the nature of heuristics whose performances are evaluated empirically, no quantum advantage with heuristics is emerging yet. This is where the second type of quantum algorithms differ: they are *exact* algorithms (i.e. that output the optimal solution with a high probability of success) that provide theoretical speed-ups for several types of problems and algorithms, as displayed by Nannicini [21] and Sutter et al. [30]. Notice that with the current quantum hardware, it is impossible to implement them today because of the huge size of quantum resources they require.

Grover [12] provides one key exact quantum algorithm, that achieves a quadratic speed-up when searching for a specific element in an unsorted table, where the complexity is computed as the number of queries of the table and is done by an oracle. Grover Search represents the routine of many exact quantum algorithms. For instance, Durr and Hoyer [7] use Grover Search as a subroutine for a hybrid quantum-classical algorithm that finds the minimum of an unsorted table, resulting in the algorithm called Quantum Minimum Finding (QMF). Later, Ambainis et al. [3] combine QMF with dynamic programming to address NP-hard vertex ordering problems, such as the Traveling Salesman Problem (TSP) or the Minimum Set Cover problem. The problems of interest must satisfy a specific property which implies that they can be solved by classical

dynamic programming in $\mathcal{O}^*(c^n)$, with c is usually not smaller than 2. Henceforth, we use $\mathcal{O}^*$ which is the usual asymptotic notation that ignores the polynomial factors. The hybrid algorithm of Ambainis et al. [3] reduces the complexity to $\mathcal{O}^*(c^n_{\text{quant}})$ for $c_{\text{quant}} < c$. As an example, Held and Karp [14] dynamic programming solves the TSP in $\mathcal{O}^*(2^n)$ whereas the hybrid algorithm of Ambainis et al. [3] achieves to solve it in $\mathcal{O}^*(1.728^n)$. Following the work of Ambainis et al. [3], other NP-hard problems have been tackled with the idea of combining Grover Search (or QMF) and classical dynamic programming. This has led to quantum speed-ups for the Steiner Tree problem (Miyamoto et al. [19]), the graph coloring problem (Shimizu and Mori [29]), and the subset sum problem (Allcock et al. [2]).

The purpose of this work is to provide a general method, a hybrid quantum-classical bounded-error algorithm, adapting the seminal idea of Ambainis et al. [3] to reduce the 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. [32] 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. A scheduling problem lies in finding the optimal assignation of a set of jobs to machines over time. Each job j is defined by at least a processing time p_j and possibly additional data like a due date d_j, a deadline $\tilde{d}_j$, or even a weight w_j reflecting its priority. One or more machines can process the set of jobs, however, at any time point, a machine can only process one job at a time. The computation of a schedule is done in order to minimize a given objective function. Throughout this paper, we use the usual notation $\alpha|\beta|\gamma$, introduced by Graham et al. [9], to describe the scheduling problem consisting of α machines, with the constraints β and the criterion γ to be minimized. For instance, $1|\tilde{d}_j|\sum_j w_j C_j$ is the problem of minimizing the total weighted completion time with deadlines constraints on a single machine. The reader interested in scheduling can refer to any textbook in scheduling, e.g. to Pinedo [23]. In this paper, we generalize the hybrid algorithm and its theoretical speed-up of Grange et al. [10], that dealt with one specific kind of recurrence property satisfied by many single-machine scheduling problem, to more general problems that satisfy other recurrence properties, consequently adapting the description of the hybrid algorithm.

Let $\mathcal{P}$ be a problem with a recurrence structure that can be solved by classical Dynamic Programming Across the Subsets (DPAS) in exponential time. For scheduling problems, it essentially

amounts to satisfying the following property: for a given set of jobs J, the optimal solution for J is the best concatenation of optimal solutions for X and $J \setminus X$ among all $X \subseteq J$ such that $|X| = |J|/2$ (modulo an additive term due to the concatenation). The main idea of our hybrid algorithm is to make use of this particular dynamic programming structure and combine it with Grover Search or QMF to reduce the exponential part of the time complexity. Notice that the dynamic programming properties that must be satisfied for our problem are sometimes more complex than those used for a classical dynamic programming resolution. In this case, a pseudo-polynomial factor can appear, depending on the formulation of these dynamic programming properties. Thus, we describe these new dynamic programming properties and give the mathematical description of the algorithm in the gate-based quantum computing model.

Problem	Our hybrid algorithm	Best classical algorithm
$1\|\tilde{d}_j\|\sum w_jC_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(2^n)$, T'kindt et al. [32]
$1\|\|\sum w_jT_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(2^n)$, T'kindt et al. [32]
$1\|prec\|\sum w_jC_j$	$\mathcal{O}^*\left(1.728^n\right)$	$\mathcal{O}^*((2-\epsilon)^n)$, for small ϵ, Cygan et al. [6]
$1\|r_j\|\sum w_jU_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(\sum w_j \cdot \sum p_j \cdot 2^n)$, Ploton and T'kindt [25]
$1\|r_j\|\sum w_jC_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot (\sum p_j)^4 \cdot 1.728^n\right)$	$\mathcal{O}^*(\sum w_j \cdot (\sum p_j)^2 \cdot 2^n)$, Ploton and T'kindt [25]
$F3\|\|C_{\max}$	$\mathcal{O}^*\left((\sum p_{ij})^4 \cdot 1.728^n\right)$	$\mathcal{O}^*(3^n)$, Shang et al. [28], Ploton and T'kindt [26]

Table 1: Comparison of worst-case time complexities between our hybrid algorithm and the best-known classical algorithm

Our contributions. We provide a bounded-error hybrid quantum-classical algorithm that extends the work of Ambainis et al. [3] to more general problems. In particular, it applies to problems with temporal constraints and non-linear objective functions found in the scheduling literature as we illustrate on several examples. Specifically, we cover three types of problems $\mathcal{P}$ that satisfy three different kinds of dynamic programming properties. Not only do we tackle problems for which the dynamic programming property is based on the *addition* of optimal values of the problem on sub-instances (as done by Grange et al. [10]) but we also consider problems for which the dynamic programming naturally applies on the *composition* of optimal values of the problem on sub-instances. Furthermore, we address the 3-machine flowshop problem that differs from previous problems by the nature of the recurrence property and widen the range of problems solved by our

hybrid algorithm. Last, we also propose an approximation algorithm for the 3-machine flowshop problem. In each case, the algorithm slightly differs depending on which case we work on, but the central idea is the same. We show that the algorithm reduces the worst-case time complexity of problems for which the current best classical resolution is by DPAS, sometimes at the cost of an additional pseudo-polynomial factor. We summarize in Table 1 the complexities on several NP-hard scheduling problems we tackle in this paper.

The structure of the paper. We detail in Section 2 the first case where $\mathcal{P}$ is related to a minimization problem (defined in Subsection 2.1), that itself includes two cases according to the nature of the dynamic programming properties it satisfies (Additive or Composed DPAS). We illustrate these notions with NP-hard single-machine scheduling problems. Next, we describe in Section 3 the hybrid quantum-classical algorithm Q-DDPAS that solves with high probability the problems of interest, recalling basic notions of quantum complexity. We end in Section 4 by adapting the previous Q-DDPAS algorithm to problem $\mathcal{P}$ that is related more naturally to a decision problem (defined in Subsection 4.1), satisfying other dynamic programming properties. We illustrate it on the 3-machine flowshop scheduling problem. Additionally, we provide an approximation scheme for the 3-machine flowshop, based on the hybrid algorithm, that disposes of the pseudo-polynomial factor in the time complexity. Appendix A recalls well-known bounds useful to derive the complexities of the algorithm while Appendix B provides a detailed proof of the correctness of our main algorithm Q-DDPAS.

2 Dynamic Programming Across the Subsets

2.1 Problems definitions

Let us describe the type of combinatorial optimization problems $\mathcal{P}$ on which DPAS can be applied. An instance of problem $\mathcal{P}$ is denoted by $\mathcal{I}$ and is described by a tuple of vectors of dimension n (the number of elements in the tuple depends on $\mathcal{P}$). We can think as an example of a single-machine scheduling problem with n jobs where the vectors specify processing times, due dates, and deadlines, among others. Any solution to $\mathcal{P}$ is a permutation of $[n] := \{1, \ldots, n\}$. We denote by

$S_{[n]}$ the set of all permutations of $[n]$. Our nominal problem can be cast as follows:

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

where $\Pi(\mathcal{I}) \subseteq S_{[n]}$ is the set of feasible permutations of $[n]$ according to given constraints and f is the objective function, which both depend on $\mathcal{I}$.

We introduce a problem P related to $\mathcal{P}$ that will be instrumental in deriving the dynamic programming recursion, satisfying the two following properties. First, an instance $\mathcal{J}$ of P is described by an instance $\mathcal{I}$ of $\mathcal{P}$ together with an additional parameter $t \in \mathbb{Z}$. Second, for an instance $\mathcal{I}$, there is $t_{\mathcal{I}} \in \mathbb{Z}$ such that solving $\mathcal{P}$ for $\mathcal{I}$ amounts to solving P for $\mathcal{J} = (\mathcal{I}, t_{\mathcal{I}})$. As we will solve P by dynamic programming based on a recurrence formula, it is convenient to define sub-instances as follows. For $\mathcal{J} = (\mathcal{I}, t)$ and $J \subseteq [n]$ we define the sub-instance of $\mathcal{J}$ associated to J by $\mathcal{J}(J) = (\mathcal{I}(J), t)$, where $\mathcal{I}(J)$ is the tuple of vectors of dimension $|J| \leq n$ obtained by considering the sub-vectors of $\mathcal{I}$ that are indexed by the components in J. Thus, we denote by $P_{\mathcal{J}}(J, t)$ the problem P defined on the sub-instance $(\mathcal{I}(J), t)$. We omit the index $\mathcal{J}$ in what follows and consider the solution of

$$P(J, t): \qquad \min_{\pi \in \Pi(J,t)} f(\pi, J, t), \tag{2}$$

where $\Pi(J, t) \subseteq S_J$ is the set of feasible permutations of J according to the given constraints and $f(., J, t)$ is the objective function. According to the above notations, the nominal problem $\mathcal{P}(\mathcal{I})$ is equivalent to problem $P_{(\mathcal{I}, t_{\mathcal{I}})}([n], t_{\mathcal{I}})$, or shortly, $P([n], t_{\mathcal{I}})$.

Henceforth, we note $\mathrm{OPT}[J, t]$ the optimal value of $P(J, t)$, for $J \subseteq [n]$ and $t \in \mathbb{Z}$. Throughout the dynamic programming recursions, we shall consider values of t that may differ from $t_{\mathcal{I}}$. More specifically, we will introduce the bounded set $T(\mathcal{I}) \subseteq \mathbb{Z}$ and consider $t \in T(\mathcal{I})$. Again, to keep concise notations, we shall omit the explicit dependency of T on $\mathcal{I}$ and consider throughout the problem $P(J, t)$, for $J \subseteq [n]$ and $t \in T$.

To illustrate the above notations, we consider $\mathcal{P}$ as a single-machine scheduling problem of n jobs. Thus, problem $P(J, t)$ is the problem P that schedules only jobs in $J \subseteq [n]$ starting at time $t \in \mathbb{Z}$. The nominal problem $\mathcal{P}$ schedules all the jobs and starts at time $t = 0$. In other words, we wish to solve $P([n], 0)$ and find the optimal value $\mathrm{OPT}[[n], 0]$.

In what follows, we distinguish between two types of recurrences satisfied by problem (2). The

first one is the recurrence that *adds* optimal values of the related problem on sub-instances, called Additive DPAS and introduced in Subsection 2.2. The second one is the recurrence that *composes* optimal values of the related problem on sub-instances, called Composed DPAS and defined in Subsection 2.3. Notice that in the first case, the recurrence applies to the related problem (2) that derive directly from $\mathcal{P}$. In that sense, it constitutes the easiest and most natural way to define the recurrence. In the second case, we find single-machine scheduling problems that necessitate going through a slightly different related problem to apply recurrence. In our examples, the use or not of this latter *auxiliary* problem is driven by the nature of the constraints. For instance, single-machine scheduling problems with deadline constraints naturally satisfy Additive DPAS recurrence whereas those with release date constraints need the Composed DPAS formulation. In both cases, we illustrate the dynamic programming properties with single-machine scheduling problems. Thus, throughout this section, we assume that we aim at solving $P([n], t_{\mathcal{I}})$, with $t_{\mathcal{I}} = 0$. Without loss of generality, other problems than scheduling problems may also be solved by the dynamic programming algorithm proposed in this work.

2.2 Additive DPAS

Let us start by defining problems for which the constraints defined by Π are compatible with the addition of optimal values of the problem on sub-instances, formally defined below. For instance, deadline constraints and precedence constraints are such constraints for single-machine scheduling problems as we illustrate after. Problem $\mathcal{P}$ can be solved by dynamic programming if the related problem P satisfies one of the two recurrences (Add-DPAS) and (Add-D-DPAS). Notice that solving $\mathcal{P}$ with our hybrid quantum-classical algorithm necessitates P to satisfy both recurrences. However, we observe in Remark 2.4 that P satisfies one if and only if it satisfies the other. Let us introduce the first recurrence. Henceforth, we denote by $2^{[n]}$ the set of all subsets of $[n] = \{1, \dots, n\}$.

Property 2.1 (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 $\mathrm{OPT}[\emptyset, t_0] = 0$.

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. Notice the presence of the additional parameter t_0 in the above property that is a constant throughout the whole recursion (Add-DPAS) and does not impact the resulting computational complexity. The use of that extra parameter defined in (2) shall be necessary later when applying our hybrid algorithm.

The previous property enables to solve problem $\mathcal{P}$ by dynamic programming with the following time complexity.

Lemma 2.2 (Additive DPAS complexity)**.** (Add-DPAS) *solves* $\mathcal{P}$ *in* $\mathcal{O}^*(2^n)$.

Proof. We solve Equation (Add-DPAS) for all J such that $|J| = k$, and for $t_0 = 0$, starting from $k = 1$ to $k = n$. For a given J, the values $\{\mathrm{OPT}[J \setminus \{j\}, 0] : j \in J\}$ are known, so $\mathrm{OPT}[J, 0]$ is computed in time $poly(n) \cdot k$ according to Equation (Add-DPAS) (the computation of g is polynomial). Eventually, the total complexity of computing $\mathrm{OPT}[[n], 0]$ is

$$\sum_{k=1}^{n} poly(n) k \binom{n}{k} = poly(n) \cdot n \cdot 2^{n-1} = \mathcal{O}^*(2^n).$$

□

The problem P related to $\mathcal{P}$ must not only satisfy recurrence (Add-DPAS) but also recurrence (Add-D-DPAS) below.

Property 2.3 (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* $t \in T$*,*

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

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

Notice that if $\mathrm{OPT}[X, t]$ for $X \subseteq [n]$ and $t \in T$ is infeasible, then by convention $\mathrm{OPT}[X, t] = +\infty$. However, differently from the previous recurrence (Add-DPAS), recurrence (Add-D-DPAS) now

calls $\mathrm{OPT}[X', t']$ for t' that may be different than t_0 by the use of the function t_{shift}. This will lead to the introduction of the pseudo-polynomial factor $|T|$ in its complexity detailed below. Next, we underline that both recurrences are essentially equivalent.

Remark 2.4. *We observe that problem* (2) *satisfies recurrence* (Add-DPAS) *if and only if it satisfies* (Add-D-DPAS). *This can be seen by developing recursively both recurrences, which essentially leads to optimization problems over $\pi \in S_{[n]}$, whose objective functions respectively involve g in the first case and h and t_{shift} in the second case. Here, one readily verifies that g can then be defined from h and t_{shift} and reciprocally.*

Despite the previous remark, the two recurrences differ on the size of the subsets considered along the recursions, leading to different formulations and therefore require more or less sub-problems to be solved optimally in the dynamic programming process. This is formalized in the following lemma.

Lemma 2.5 (Additive Dichotomic DPAS complexity)**.** (Add-D-DPAS) *solves $\mathcal{P}$ in $\omega(|T| \cdot 2^n)$.*

Proof. First, we note that to solve $\mathcal{P}$ with (Add-D-DPAS), n must be a power of 2. If this is not the case, we can always transform the instance such that we fall back into the previous case. Thus, without loss of generality, we suppose that $n = 2^N$ for $N \in \mathbb{N}$. We solve Equation (Add-D-DPAS) for all J such that $|J| = 2^k$, and for all $t \in T$, starting from $k = 1$ to $k = N$. For a given J, the values $\{\mathrm{OPT}[X, t'] : X \subseteq J \text{ s.t. } |X| = |J|/2\,, t' \in T\}$ are known, so $\mathrm{OPT}[J, t]$ is computed in time $poly(n)\binom{2^k}{2^{k-1}}$ according to Equation (Add-D-DPAS) (the computation of t_{shift} and h is polynomial). Thus, computing all $\mathrm{OPT}[J, t]$ for any J of size 2^k and $t \in T$ is done in time $|T|poly(n)\binom{2^k}{2^{k-1}}\binom{n}{2^k}$. Eventually, the total complexity is equal to

$$C(n) = |T|poly(n)\sum_{k=1}^{N}\binom{2^k}{2^{k-1}}\binom{n}{2^k}.$$

Second, we compute the complexity of (Add-D-DPAS). For that, we consider the sequence $(C(2^i))_{i\in\mathbb{N}}$, knowing that for families of instances with a size different from a power of 2, we transform them artificially into families of instances of size of the following power of 2. Let $n = 2^i$

for $i \in \mathbb{N}$. A lower bound on $C(n)$ is the sum of the two last terms:

$$C(n) > |T|poly(n)\left(\binom{n}{n/2} + \binom{n}{n/2}\binom{n/2}{n/4}\right) \approx A|T|poly(n)\frac{2^{1.5n}}{n},$$

where A is a constant. The asymptotic equivalent is readily obtained with the Stirling equivalent for factorials, $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ for $n \in \mathbb{N}$. Thus, C dominates asymptotically $n \mapsto |T| \cdot 2^n$. In other words, $C(n) = \omega(|T| \cdot 2^n)$. □

Lemma 2.5 shows that solving $\mathcal{P}$ using classical dynamic programming with recurrence (Add-D-DPAS) is worse than with (Add-DPAS). It explains why the dynamic programming algorithms found in the literature (e.g. in T'kindt et al. [32]) rely on recurrence (Add-DPAS). In the next section, we describe a hybrid algorithm we call Quantum Dichotomic DPAS (Q-DDPAS) that improves the complexity of solving $\mathcal{P}$ by combining recurrence (Add-DPAS) and recurrence (Add-D-DPAS) with Grover Search. Before introducing this algorithm, we illustrate the two recurrences (Add-DPAS) and (Add-D-DPAS) above with several examples of single-machine scheduling problems that can, consequently, be solved by Q-DDPAS. Let us begin with the NP-hard scheduling problem with deadline constraints and minimization of the total weighted completion time. In all the scheduling examples that follow, we note $p(J) := \sum_{j \in J} p_j$, for $J \subseteq [n]$.

Example 1 (Minimizing the total weighted completion time with deadlines, $1|\tilde{d}_j|\sum_j w_j C_j$)**.** *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 $p([n]) = \sum_{i=1}^{n} p_i$. We define the related problem P 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, and not 0 as usual. Our problem of interest $\mathcal{P}$ is $P([n], 0)$, and it can

be solved by (Add-DPAS). *Indeed, Equation* (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). This problem* P *also satisfies* (Add-D-DPAS). *Indeed, Equation* (Add-D-DPAS) *is valid for the following functions:*

$\forall X \subseteq J \subseteq [n] : |X| = |J|/2, \forall t \in T$,

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

initialized by, for $j \in [n]$ *and* $t \in T$,

$$\mathrm{OPT}[\{j\},t] = \begin{cases} w_j(p_j + t) & \text{if } \tilde{d}_j \geq p_j + t \\ +\infty & \text{otherwise} \end{cases}$$

We present another example which is the strongly NP-hard scheduling problem with minimization of the total weighted tardiness.

Example 2 (Minimizing the total weighted tardiness, $1||\sum_j w_j T_j$)**.** *For each job* $j \in [n]$*, we are given a weight* w_j*, a processing time* p_j*, and a due date* d_j*. Let* $T = [\![0, p([n])]\!]$*. We define the related problem* P *as follows: for* $J \subseteq [n]$ *and* $t \in T$,

$$\Pi(J,t) = S_J,$$

and for $\pi \in \Pi(J,t)$*:*

$$f(\pi,J,t) = \sum_{j \in J} w_j \max(0, C_j(\pi) - d_j + t)\,,$$

where $\max(0, C_j - d_j + t)$ *represents the tardiness of job* j *for the* effective due date $d_j - t$*. Our problem of interest* $\mathcal{P}$ *is* $P([n], 0)$ *, and it can be solved by* (Add-DPAS). *Indeed, Equation* (Add-DPAS) *is valid with:* $\forall J \subseteq [n], \forall j \in J, \forall t \in T$,

$$g(J,j,t) = w_j \max(0, p(J \cup \{j\}) - d_j + t)\,,$$

where the computation of g is polynomial (linear). This problem also satisfies (Add-D-DPAS). *Indeed, Equation* (Add-D-DPAS) *is valid for the following functions:*

$\forall X \subseteq J \subseteq [n]$ *s.t.* $|X| = |J|/2, \forall t \in T$,

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

initialized by, for $j \in [n]$ and $t \in T$,

$$\mathrm{OPT}[\{j\}, t] = w_j \max(0, p_j - d_j + t)\,.$$

Eventually, we consider the scheduling problem with precedence constraints and minimization of the total weighted completion time that is also NP-hard. Conversely to the two previous examples, the set T is reduced to $\{0\}$ and function h translates the potential infeasibility of the concatenation of problem P on two sub-instances.

Example 3 (Minimizing the total weighted completion time with precedence constraints, $1|prec|\sum_j w_j C_j$)**.** *We are given, for each job $j \in [n]$, a processing time p_j, a weight w_j, and a set of precedence constraints $K = \{(i, j) : i \prec j\}$ that contains all pairs of jobs (i, j) such that i precedes j. Let $T = \{0\}$. Here, an instance of the problem P under consideration is only indexed by the chosen subset of $[n]$. Thus, we consider the problem P as follows: for $J \subseteq [n]$,*

$$\Pi(J, 0) = \{\pi \in S_J \,|\, \pi \text{ respects } K\}\,,$$

and for $\pi \in \Pi(J, 0)$:

$$f(\pi, J, 0) = \sum_{j \in J} w_j C_j(\pi)\,.$$

Our problem of interest $\mathcal{P}$ is $P([n], 0)$, and it can be solved by (Add-DPAS). *Indeed, Equation* (Add-DPAS) *is valid for:*

$$\forall J \subseteq [n], \forall j \in J, \quad g(J, j, 0) = \begin{cases} +\infty & \text{if } \exists (j, k) \in E | k \in J \\ w_j p(J \cup \{j\}) & \text{otherwise} \end{cases}$$

where the computation of g is polynomial (quadratic). This problem P also satisfies (Add-D-DPAS).

Indeed, Equation (Add-D-DPAS) *is valid for the following functions:* $\forall X \subseteq J \subseteq [n]$ *such that* $|X| = |J|/2$,

$$t_{shift}(J, X, 0) = 0$$

$$h(J, X, 0) = \begin{cases} +\infty & \text{if } \exists (j,k) \in E | j \in J \setminus X \text{ and } k \in X \\ p(X) \cdot \sum_{j \in J \setminus X} w_j & \text{otherwise} \end{cases}$$

where the computation of h *is polynomial (quadratic). The initialization is, for* $j \in [n]$,

$$\mathrm{OPT}[\{j\}, 0] = w_j p_j \,.$$

2.3 Composed DPAS

In this subsection, we study problems whose constraints do not enable the computation of an optimal solution by simply adding optimal values of sub-instances as for Additive DPAS (see Subsection 2.2). Instead, we consider that the optimal value of $\mathcal{P}$ can be computed with the composition of optimal values of a slightly different related minimization problem P' on sub-instances. We call this problem an *auxiliary* problem. To illustrate, we consider single-machine scheduling problems with release date constraints. The recurrence formulas below are inspired by the work of Lawler [17] for the problem $1|r_j, pmtn| \sum w_j U_j$, namely minimizing the total weighted number of late jobs on a single machine under preemption and release date constraints.

2.3.1 Auxiliary problems definitions

Let us introduce the set of values of the objective function of $\mathcal{P}$,

$$E(\mathcal{I}) = \{ f(\pi, \mathcal{I}) : \pi \in \Pi(\mathcal{I})) \} \,.$$

We define the auxiliary problem $\mathcal{P}'$ as follows. An instance $\mathcal{I}'$ of $\mathcal{P}'$ is defined as an instance $\mathcal{I}$ of $\mathcal{P}$ together with an additional parameter $\epsilon \in E(\mathcal{I})$, so $\mathcal{P}'$ reads

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

where $\mathcal{I}' = (\mathcal{I}, \epsilon)$, $\Pi'(\mathcal{I}') \subseteq S_{[n]}$ is a set of feasible permutations of $[n]$ according to $\mathcal{I}$ and ϵ and f' is the objective function. In what follows, the objective of $\mathcal{P}$ is to find the smallest $\epsilon \in E(\mathcal{I})$ such that the associated problem $\mathcal{P}'$ is bounded, formally

$$\mathcal{P}(\mathcal{I}): \quad \min_{\epsilon \in E} \Big\{ \epsilon : \min_{\pi \in \Pi'(\mathcal{I}, \epsilon)} f'(\pi, (\mathcal{I}, \epsilon)) < +\infty \Big\}. \tag{3}$$

We introduce a problem P' related to $\mathcal{P}'$ exactly as it has been done for P related to $\mathcal{P}$ in Subsection 2.1. Thus, an instance $\mathcal{J}'$ of P' is an instance $\mathcal{I}'$ of $\mathcal{P}'$ with an additional parameter $t \in T(\mathcal{I}')$, so $\mathcal{J}' = (\mathcal{I}, t, \epsilon)$. As before, we consider the sub-instance $\mathcal{J}'(J) = (\mathcal{I}(J), t, \epsilon)$ corresponding to $J \subseteq [n]$, and consider the solution of

$$P'(J, t, \epsilon): \qquad \min_{\pi \in \Pi'(J, t, \epsilon)} f'(\pi, J, t, \epsilon), \tag{4}$$

where $\Pi'(J, t, \epsilon) \subseteq S_J$ is the set of feasible permutations of J according to the given constraints and $f'(., J, t, \epsilon)$ is the objective function. We note $\mathrm{OPT}[J, t, \epsilon]$ the optimal value of $P'(J, t, \epsilon)$, for $J \subseteq [n]$ and $t, \epsilon \in \mathbb{Z}$. As in Subsection 2.1, we introduce the bounded set $T(\mathcal{I}') \subseteq \mathbb{Z}$ and consider parameters $t \in T(\mathcal{I}')$. Once again, we omit the dependency to $\mathcal{I}'$ to lighten the notations, denoting by T, respectively E, the set $T(\mathcal{I}')$, respectively $E(\mathcal{I}')$. Rewriting (3) with the above notations yields

$$\mathcal{P}: \quad \min_{\epsilon \in E} \Big\{ \epsilon : \mathrm{OPT}[[n], 0, \epsilon] < +\infty \Big\}. \tag{5}$$

Thus, solving $\mathcal{P}$ amounts to solving $P'([n], 0, \epsilon)$ for all $\epsilon \in E$. Observe that the optimal solution of $\mathcal{P}$ is the optimal solution of $P'([n], 0, \epsilon^*)$ for ϵ^* the optimal value of $\mathcal{P}$.

Notice that we consider an extra parameter in E to define our auxiliary problem as generic as possible. This is motivated by the fact that it enables to tackle strongly NP-hard problems as shown in the examples that end this subsection. However, this formulation covers the *simple* case of $\mathcal{P}$ deriving in a problem (not involving an auxiliary problem) that satisfies the composition of optimal values on sub-instances, essentially by setting $E = \emptyset$.

Remark 2.6. *We formulate $\mathcal{P}$ as in* (5) *because in the examples we study, finding the optimal value amounts to finding the smallest ϵ such that the auxiliary problem $P'(J, t, \epsilon)$ admits a* feasible

solution. However, all the results of this subsection generalize to any problem formulated as

$$\min_{\epsilon \in E} \Big\{ \epsilon : bool(\mathrm{OPT}[[n], 0, \epsilon]) = true \Big\},$$

where bool is a boolean function.

2.3.2 Composed DPAS recurrences

To solve the nominal problem $\mathcal{P}$ by classical dynamic programming, problem P' must satisfy recurrence (Comp-DPAS) or recurrence (Comp-D-DPAS) below (as in Remark 2.4, we can state that a problem satisfies one if and only if it satisfies the other one). As we explain later, solving $\mathcal{P}$ with our hybrid algorithm necessitates problem P' to satisfy the two recurrences.

Property 2.7 (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 $\epsilon \notin E$.*

Recurrence (Comp-DPAS) differs from recurrence (Add-DPAS) in two aspects. First, the optimal values of the problem on sub-instances are composed, and not added, because of the nature of the constraints. Second, the search for the minimum value is done not only over all jobs in J, but also over all values in E. More precisely, for a given $\epsilon_0 \in E$, the optimal value of $P'(J, t, \epsilon_0)$ is the minimum value of all possible composition of optimal values of the problem on sub-instances with parameters ϵ_1 and ϵ_2 such that $\epsilon_1 + \epsilon_2 = \epsilon_0$. We have the following results.

Lemma 2.8 (Composed DPAS complexity)**.** *Let $\epsilon_0 \in E$.* (Comp-DPAS) *solves $P'([n], 0, \epsilon_0)$ in $\mathcal{O}^*(|E|^2 \cdot |T| \cdot 2^n)$.*

Proof. This proof is essentially the same as the one of Lemma 2.2. To compute $\mathrm{OPT}[[n], 0, \epsilon_0]$, we need to solve Equation (Comp-DPAS) for all J such that $|J| = k$ starting from $k = 1$ to $k = n$, and for all $t \in T$ and $\epsilon \in E$. For a given J, $t \in T$ and $\epsilon \in E$, the values $\{\mathrm{OPT}[J \setminus \{j\}, t', \epsilon'] : j \in J, t' \in T, \epsilon' \in E\}$ and $\{\mathrm{OPT}[\{j\}, t', \epsilon'] : j \in J, t' \in T, \epsilon' \in E\}$ are known, so $\mathrm{OPT}[J, t, \epsilon]$ is computed in time $|E| \cdot k$ according to Equation (Comp-DPAS). Thus, computing all $\mathrm{OPT}[J, t, \epsilon]$ for any J of

size k and for $t \in T$ and $\epsilon \in E$ is done in time $|T| \cdot |E|^2 \cdot k\binom{n}{k}$. Eventually, the total complexity of computing $\mathrm{OPT}[[n], 0, \epsilon_0]$ is

$$\sum_{k=1}^{n} |T| \cdot |E|^2 \cdot k\binom{n}{k} = \mathcal{O}^*(|T| \cdot |E|^2 \cdot 2^n).$$

□

Corollary 2.9 (Composed DPAS complexity)**.** (Comp-DPAS) *solves* $\mathcal{P}$ *in* $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 2^n)$.

Proof. Solving $\mathcal{P}$ amounts to solving $P'([n], 0, \epsilon)$ for all $\epsilon \in E$, according to (5). The complexity results directly from the previous Lemma 2.8. □

The auxiliary problem P' must satisfy the following recurrence (Comp-D-DPAS) in addition to recurrence (Comp-DPAS).

Property 2.10 (Composed Dichotomic DPAS)**.** *For all* $J \subseteq [n]$ *of even cardinality,* $t \in T$ *and* $\epsilon \in E$*,*

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

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

Lemma 2.11 (Composed Dichotomic DPAS complexity)**.** *Let* $t_0 \in T$ *and* $\epsilon_0 \in E$*.* (Comp-D-DPAS) *solves* $P'([n], t_0, \epsilon_0)$ *in* $\omega(|E|^2 \cdot |T| \cdot 2^n)$*.*

Proof. This proof is essentially the same as the one of Lemma 2.5 with the same modifications that for the proof of Lemma 2.8. □

Corollary 2.12 (Composed Dichotomic DPAS complexity)**.** (Comp-D-DPAS) *solves* $\mathcal{P}$ *in* $\omega(|E|^3 \cdot |T| \cdot 2^n)$*.*

Proof. See proof of Corollary 2.9. □

As for the Additive DPAS, we notice that, with a classical dynamic programming algorithm, the time complexity to solve $\mathcal{P}$ with recurrence (Comp-DPAS) is better than with recurrence (Comp-D-DPAS). We show that our hybrid algorithm Q-DDPAS improves these complexities. But first, we illustrate the recurrences (Comp-DPAS) and (Comp-D-DPAS) with two single-machine

scheduling examples. Let us first consider the strongly NP-hard problem of minimizing the total weighted sum of late jobs with release date constraints.

Example 4 (Minimizing the total weighted number of late jobs with release date constraints, $1|r_j|\sum w_jU_j$)**.** *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. Let $T = [\![0, \sum_{j=1}^{n} p_j]\!] \cup \{+\infty\}$ and $E = [\![0, \sum_{j=1}^{n} w_j]\!]$. For a given $\epsilon \in E$, we consider the problem P' as follows: $\forall J \subseteq [n], t \in T$,*

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

where C_{max} is the makespan, 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. If j is late for the permutation π, $U_j(\pi) = 1$, otherwise, $U_j(\pi) = 0$. In other words, $\mathrm{OPT}[J,t,\epsilon]$ *is the minimum makespan for jobs in J beginning at time t where exactly ϵ jobs are late. Notice that if there is no solution where ϵ jobs are late, then* $\mathrm{OPT}[J,t,\epsilon] = +\infty$*. Our problem of interest is* $\mathcal{P} = \min_{\epsilon \in E}\{\epsilon : \mathrm{OPT}[[n],0,\epsilon] < +\infty\}$.

Problem P' satisfies both (Comp-DPAS) *and* (Comp-D-DPAS) *recurrences. The initialization of the two recurrences is, for $j \in [n]$, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[\{j\},t,\epsilon] = \begin{cases} \underbrace{\max(t,r_j) + p_j}_{C_j}, & \text{if } C_j \leq d_j \text{ and } \epsilon = 0 \\ +\infty, & \text{if } C_j > d_j \text{ and } \epsilon = 0 \\ +\infty, & \text{if } C_j \leq d_j \text{ and } \epsilon = w_j \\ C_j, & \text{if } C_j > d_j \text{ and } \epsilon = w_j \\ +\infty, & \text{if } \epsilon \in [\![1, w_j - 1]\!] \cup [\![w_j + 1, \sum_{k=1}^{n} w_k]\!] \end{cases}$$

Notice that for this example, Equation (Comp-DPAS) *can be simplified to*

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

With this particular recurrence formula, solving $\mathcal{P}$ with (Comp-DPAS) *is done in $\mathcal{O}^*(|E|^2 \cdot |T| \cdot 2^n)$ instead of $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 2^n)$.*

As we mentioned before, (Comp-DPAS) and (Comp-D-DPAS) recurrences naturally apply to scheduling problems with release date constraints. We illustrate that with another example, which is the strongly NP-hard problem of minimizing the total weighted completion time with release date constraints.

Example 5 (Minimizing the total weighted completion time with release date constraints, $1|r_j|\sum w_jC_j$)**.** *Each job $j \in [n]$ has a weight w_j, a processing time p_j, and a release date r_j. Let $T = [\![0, \sum_{j=1}^{n} p_j]\!] \cup \{+\infty\}$ and $E = [\![0, \sum_{j=1}^{n} w_j \cdot \sum_{j=1}^{n} p_j]\!]$. For a given $\epsilon \in E$, we consider the problem P' as follows: $\forall J \subseteq [n], t \in T$,*

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

where C_{max} is the makespan, and

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

where C_j is the completion time of job j. In other words, $\mathrm{OPT}[J,t,\epsilon]$ *is the minimum makespan for jobs in J beginning at time t where the weighted completion time is exactly ϵ. Notice that if there is no solution where this total sum is equal to ϵ, then* $\mathrm{OPT}[J,t,\epsilon] = +\infty$*. With these notations, our problem of interest is $\mathcal{P} = \min_{\epsilon \in E}\{\epsilon : \mathrm{OPT}[[n],0,\epsilon] < +\infty\}$.*

We can note that problem P' satisfies the two recurrences (Comp-DPAS) *and* (Comp-D-DPAS)*. The initialization of the recurrences is, for $j \in [n]$, $t \in T$ and $\epsilon \in E$,*

$$\mathrm{OPT}[\{j\},t,\epsilon] = \begin{cases} \underbrace{\max(t,r_j) + p_j}_{C_j}, & \text{if } \epsilon = w_jC_j \\ +\infty, & \text{otherwise} \end{cases}$$

3 Hybrid algorithm Q-DDPAS

In this section, we describe in the gate-based quantum computing model our algorithm Q-DDPAS that applies to any problem that satisfies the recurrences properties described in Section 2. But

first, we introduce some preliminary notions on quantum circuits and notations for the description of Q-DDPAS.

3.1 Preliminaries

Let us begin with some notions of time complexity for quantum circuits.

Definition 3.1. *Let us consider a family of quantum circuits $(\mathcal{Q}_n)_{n\in\mathbb{N}}$ of complexity $\mathcal{O}(C(n))$, meaning that $\mathcal{Q}_n$ is a circuit that applies on n qubits and contains $f(n)$ universal quantum gates, where $f(n) = \mathcal{O}(C(n))$. This family is* efficient *if $C(n) = n^\alpha$ for $\alpha > 0$.*

Observation 3.2 (Complexity of quantum circuits (Nielsen and Chuang [22]))**.** *Let U_1 and U_2 be two quantum circuits, with complexity $\mathcal{O}(C_1(n))$ and $\mathcal{O}(C_2(n))$, respectively. The complexity of the composition $U_1 \cdot U_2$ is*

$$\mathcal{O}(C_1(n) + C_2(n)) = \mathcal{O}(\max(C_1(n), C_2(n)))\,.$$

The tensor product $U_1 \otimes U_2$ has the same complexity.

Observation 3.3 (Classical algorithm into quantum circuit (Bennett [4]))**.** *Any classical algorithm $\mathcal{A}$ can be described as a quantum circuit $U_\mathcal{A}$. The complexity of $U_\mathcal{A}$ is equal to the complexity of $\mathcal{A}$.*

Before we describe specific sets and quantum circuits in the following subsections, we introduce the generic quantum circuit associated with the Quantum Minimum Finding (QMF) algorithm of Durr and Hoyer [7], that constitutes a fundamental subroutine in our algorithm. This algorithm essentially applies several times the search algorithm of Grover [12].

Definition 3.4 (Circuit U_{QMF})**.** *Let $f : [n] \to \mathbb{Z}$ be a function and let U_f be its corresponding quantum circuit, specifically,*

$$U_f \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left|f(i)\right\rangle, \quad \forall i \in [n]\,.$$

We note $U_{\mathrm{QMF}}[U_f]$ the quantum circuit corresponding to the Quantum Minimum Finding algorithm of Durr and Hoyer [7] that computes with high probability the minimum value of f and the corresponding minimizer:

$$U_{\mathrm{QMF}}[U_f] \sum_{i=1}^{n} \frac{1}{\sqrt{n}} \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle = \sum_{i=1}^{n} \frac{1}{\sqrt{n}} \left|i\right\rangle \left|\arg\min_{i\in[n]}\{f(i)\}\right\rangle \left|\min_{i\in[n]}\{f(i)\}\right\rangle .$$

Observation 3.5 (Complexity of U_{QMF})**.** *The complexity of the Quantum Minimum Finding algorithm is* $\mathcal{O}\left(\sqrt{n} \cdot C_f(n)\right)$*, where* n *is the size of the domain of* f *and* $\mathcal{O}(C_f(n))$ *is the complexity of the circuit* U_f*. Thus, the complexity of* $U_{\text{QMF}}[U_f]$ *is*

$$\mathcal{O}\left(\sqrt{n} \cdot C_f(n)\right)$$

according to Observation 3.3.

Remark 3.6 (Success probability and bounded-error algorithm)**.** *Durr and Hoyer [7] prove that QMF finds the minimum value with a probability of success bigger than* $\frac{1}{2}$*. Thus, for* $\epsilon > 0$*, finding the minimum value with probability* $1 - \epsilon$ *is achieved by repeating* $\mathcal{O}(1)$ *times QMF. Henceforth, we refer to this statement when we write that QMF finds the minimum value* with high probability*. Equivalently, we say that this is a* bouded-error *algorithm.This notion is defined by Nielsen and Chuang [22].*

We introduce in the two following subsections the sets and quantum circuits that constitute the building blocks of our algorithm Q-DDPAS, and we provide for each of them their complexity. Depending on the tackled problem $\mathcal{P}$ solved by the hybrid algorithm, these sets, respectively quantum circuits, slightly differ whether the related problem P satisfies (Add-DPAS) and (Add-D-DPAS), or the related auxiliary problem P' satisfies (Comp-DPAS) and (Comp-D-DPAS). In both cases, we define two sets Λ_{add} and Ω_{add} indexed by (J, t) for $J \subseteq [n]$ and $t \in T$, respectively Λ_{comp} and Ω_{comp} indexed by (J, t, ϵ) for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$. Essentially, the set $\Lambda_{\text{add}}(J, t)$ contains all the possible balanced bi-partitions of J and the associated parameter value of t_{shift}. The second set $\Omega_{\text{add}}(J, t)$ contains the optimal solutions for each bi-partition in $\Lambda_{\text{add}}(J, t)$. Similarly, the set $\Lambda_{\text{comp}}(J, t, \epsilon)$ contains all the possible balanced bi-partitions of J and the possible parameter values of T and E. The second set $\Omega_{\text{comp}}(J, t, \epsilon)$ contains the optimal solutions for each bi-partition and parameter values in $\Lambda_{\text{comp}}(J, t, \epsilon)$.

3.1.1 Additive DPAS sets and quantum circuits

Let us begin with the sets and related quantum circuits useful to the description of our algorithm for solving problems whose related problem P satisfies recurrences (Add-DPAS) and (Add-D-DPAS).

Definition 3.7 (Sets Λ_{add} and Ω_{add})**.** *For $J \subseteq [n]$ such that $|J|$ is even and for $t \in T$, we define the set*

$$\Lambda_{add}(J,t) = \left\{ (X, t, J \setminus X, t_{shift}(J,X,t)) : X \subseteq J, |X| = \frac{|J|}{2} \right\},$$

and the set

$$\Omega_{add}(J,t) = \left\{ (X, \mathrm{OPT}[X,t], J \setminus X, \mathrm{OPT}[J \setminus X, t_{shift}(J,X,t)], t) : X \subseteq J, |X| = \frac{|J|}{2} \right\}.$$

The two following quantum circuits $U_{\Lambda_{\text{add}}}$ and $U_{\Omega_{\text{add}}}$ amount, respectively, to put into uniform superposition the elements of Λ_{add} and Ω_{add}.

Definition 3.8 (Circuit $U_{\Lambda_{\text{add}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $t \in T$, we define $U_{\Lambda_{add}}$ as follows:*

$$U_{\Lambda_{add}} |J\rangle |t\rangle |0\rangle^{\otimes 6} =$$
$$|J\rangle |t\rangle \sum_{(\lambda_1^s, \lambda_1^t, \lambda_2^s, \lambda_2^t) \in \Lambda_{add}(J,t)} \frac{1}{\sqrt{|\Lambda_{add}(J,t)|}} |\lambda_1^s\rangle \left|\lambda_1^t\right\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^t\right\rangle |0\rangle .$$

Observe that we index the objects that represent sets by s, and the objects that represent scalars by t, because these are equal to the values in T.

Proposition 3.9 (Complexity of $U_{\Lambda_{\text{add}}}$)**.** *The complexity of $U_{\Lambda_{add}}$ is polynomial in the size of the input.*

Proof. First, let us prove that, for a given $J \subseteq [n]$ of size m for m even, the construction of the quantum superposition of subsets of J of size $m/2$ (i.e. superposition of balanced bi-partitions) is polynomial.

Let $J \subseteq [n]$ be of size m, for m even. We note $\sigma_{\text{enum}} : J \mapsto [\![1, m]\!]$ the bijection that enumerates the elements of J. We note $\sigma_{\text{bipart}} : [\![1, \binom{m}{m/2}]\!] \mapsto \left\{ (A, [\![1,m]\!] \setminus A) : |A| = \frac{m}{2} \right\}$ the bijection that enumerates the balanced bi-partitions of $[\![1, m]\!]$. Let $U_{\sigma_{\text{bipart}}}$ be the quantum circuit corresponding to the function σ_{bipart}. Specifically, for $i \in [\![1, \binom{m}{m/2}]\!]$,

$$U_{\sigma_{\text{bipart}}} |i\rangle |0\rangle |0\rangle = |i\rangle \underbrace{|A_i\rangle |[\![1,m]\!] \setminus A_i\rangle}_{\sigma_{\text{bipart}}(i)} .$$

Let $U_{\sigma_{\text{enum}}^{-1}}$ be the quantum circuit corresponding to the inverse of the function σ_{enum}. Thus,

$$U_{\sigma_{\text{enum}}^{-1}} |i\rangle |A_i\rangle |[\![1,m]\!] \setminus A_i\rangle = |i\rangle |X_i\rangle |J \setminus X_i\rangle \ ,$$

for $X_i = \sigma_{\text{enum}}^{-1}(A_i) \subseteq J$. We denote by $\sigma_{\text{enum}}^{-1}(S)$, for S a set, the operation of applying $\sigma_{\text{enum}}^{-1}$ to each element of S.

Consequently, we get a quantum superposition of all balanced bi-partitions of J by applying first $U_{\sigma_{\text{bipart}}}$ then $U_{\sigma_{\text{enum}}^{-1}}$ to a quantum register that represents the superposition of all elements in $[\![1, \binom{m}{m/2}]\!]$. For that, we require $n_q := \lceil \log_2(\binom{m}{m/2}) \rceil = \mathcal{O}(m)$ qubits, each one initially in state $|0\rangle$ on which we apply the Hadamard gate. Specifically,

$$\begin{aligned} U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} H^{\otimes n_q} |0\rangle^{\otimes n_q} |0\rangle |0\rangle &= U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |0\rangle |0\rangle \\ &= U_{\sigma_{\text{enum}}^{-1}} \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |A_i\rangle |[\![1,m]\!] \setminus A_i\rangle \\ &= \sum_{i=1}^{\binom{m}{m/2}} |i\rangle |X_i\rangle |J \setminus X_i\rangle \, . \end{aligned}$$

Let us compute the complexity of $U_{\sigma_{\text{enum}}^{-1}} U_{\sigma_{\text{bipart}}} H^{\otimes n_q}$. For a given i, computing $\sigma_{\text{bipart}}(i)$, respectively $\sigma_{\text{enum}}^{-1}(i)$, is polynomial in m. According to Observation 3.3, the complexity of $U_{\sigma_{\text{bipart}}}$, respectively $U_{\sigma_{\text{enum}}^{-1}}$, is polynomial in m. Thus, the construction of the superposition of balanced bi-partitions of J is polynomial.

Eventually, the computation of the function t_{shift} is polynomial. Thus, the complexity of $U_{\Lambda_{\text{add}}}$ is polynomial. □

Definition 3.10 (Circuit $U_{\Omega_{\text{add}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, and for $t \in T$, we define $U_{\Omega_{add}}$ as follows:*

$$U_{\Omega_{add}} |J\rangle |t\rangle |0\rangle = |J\rangle |t\rangle \sum_{\omega \in \Omega_{add}(J,t)} \frac{1}{\sqrt{|\Omega_{add}(J,t)|}} |\omega\rangle \ .$$

Proposition 3.11 (Complexity of $U_{\Omega_{\text{add}}}$)**.** *Let J be the input set. If we suppose to have stored in the QRAM the values* $\text{OPT}[X,t]$ *for all $X \subseteq J$ such that $|X| = |J|/2$ and for all $t \in T$, the complexity of $U_{\Omega_{add}}$ is polynomial in the size of the input.*

Proof. The proof follows essentially the same lines as the proof of Property 3.9. The quantum superposition of subsets is done in polynomial time, and instead of computing t_{shift}, we get values in the QRAM in constant time. □

We end this subsection with the definition of the quantum circuit of the addition required for recurrence (Add-D-DPAS).

Definition 3.12 (Circuit U_a)**.** *We define the antecedent set* $S_a = 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T$*. Let* $a : S_a \to \mathbb{Z} \cup \{+\infty\}$ *be the function:*

$$a(\omega_1^s, \omega_1^v, \omega_2^s, \omega_2^v, \omega^t) = \omega_1^v + \omega_2^v + h(\omega_1^s \cup \omega_2^s, \omega_1^s, \omega^t)\,.$$

We note U_a *the quantum circuit corresponding to* a*, namely:*

$$\forall(\omega_1^s, \omega_1^v, \omega_2^s, \omega_2^v, \omega^t) \in S_a, \quad U_a \left|\omega\right\rangle \left|0\right\rangle = \left|\omega\right\rangle \left|a(\omega)\right\rangle\,,$$

where $\left|\omega\right\rangle = \left|\omega_1^s\right\rangle \left|\omega_1^v\right\rangle \left|\omega_2^s\right\rangle \left|\omega_2^v\right\rangle \left|\omega^t\right\rangle$ *is encoded in five registers. Notice that we index the objects that represent numerical values by* v*.*

Notice that according to recurrence (Add-D-DPAS), the function a applies on objects of $\Omega_{\text{add}}(J, t)$ for $J \subseteq [n]$ and $t \in T$, explaining the choice of the antecedent set.

Proposition 3.13 (Complexity of U_a)**.** *The complexity of* U_a *is polynomial in the size of the input.*

Proof. The computation of h is polynomial (see (Add-D-DPAS)). It implies that the computation of a is polynomial, and thus U_a has a polynomial complexity (see Observation 3.3). □

Remark 3.14. *Notice that for* $J \subseteq [n]$ *and* $t \in T$*,*

$$\text{OPT}[J, t] = \min_{\omega \in \Omega_{add}(J,t)} a(\omega)\,. \tag{6}$$

3.1.2 Composed DPAS sets and quantum circuits

In this subsection, we define the sets and their associated quantum circuits used for the description of the hybrid algorithm that solves problems whose related auxiliary problem satisfies recurrences (Comp-DPAS) and (Comp-D-DPAS).

Definition 3.15 (Sets Λ_{comp} and Ω_{comp})**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and for $\epsilon \in E$, we define the set*

$$\Lambda_{comp}(J,t,\epsilon) = \left\{ (X, t_i, \epsilon_i, J \setminus X, t, \epsilon - \epsilon_i) : X \subseteq J, |X| = \frac{|J|}{2}, \epsilon_i \in E, t_i \in T \right\},$$

and the set

$$\Omega_{comp}(J,t,\epsilon) = \left\{ \begin{array}{r} (X, \mathrm{OPT}_{\epsilon_i}[X,t_i], t_i, \epsilon_i, J \setminus X, \mathrm{OPT}_{\epsilon-\epsilon_i}[J \setminus X, t], t, \epsilon - \epsilon_i) : \\ X \subseteq J, |X| = \frac{|J|}{2}, \epsilon_i \in E, t_i \in T \end{array} \right\}.$$

Definition 3.16 (Circuit $U_{\Lambda_{\text{comp}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and for $\epsilon \in E$, we define $U_{\Lambda_{comp}}$ as follows:*

$$U_{\Lambda_{comp}} |J\rangle |t\rangle |0\rangle^{\otimes 8} = \\ |J\rangle |t\rangle \sum_{(\lambda_1^s,\lambda_1^t,\lambda_1^e,\lambda_2^s,\lambda_2^t,\lambda_2^e) \in \Lambda_{comp}(J,t,\epsilon)} \frac{1}{\sqrt{|\Lambda_{comp}(J,t)|}} |\lambda_1^s\rangle \left|\lambda_1^t\right\rangle |\lambda_1^e\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^t\right\rangle |\lambda_2^e\rangle |0\rangle .$$

Observe that we index the objects that represent sets by s, the objects that represent scalars in T by t, and the objects that represent parameter values in E by e.

Proposition 3.17 (Complexity of $U_{\Lambda_{\text{comp}}}$)**.** *The complexity of $U_{\Lambda_{comp}}$ is polynomial in the size of the input.*

Definition 3.18 (Circuit $U_{\Omega_{\text{comp}}}$)**.** *For $J \subseteq [n]$ such that $|J|$ is even, for $t \in T$ and $\epsilon \in E$, we define $U_{\Omega_{comp}}$ as follows:*

$$U_{\Omega_{comp}} |J\rangle |t\rangle |\epsilon\rangle |0\rangle = |J\rangle |t\rangle |\epsilon\rangle \sum_{\omega \in \Omega_{comp}(J,t,\epsilon)} \frac{1}{\sqrt{|\Omega_{comp}(J,t,\epsilon)|}} |\omega\rangle .$$

Proposition 3.19 (Complexity of $U_{\Omega_{\text{comp}}}$)**.** *Let J be the input set. If we suppose to have stored in the QRAM the values $\mathrm{OPT}_\epsilon[X,t]$ for all $X \subseteq J$ such that $|X| = |J|/2$, for all $t \in T$ and for all $\epsilon \in E$, the complexity of $U_{\Omega_{comp}}$ is polynomial in the size of the input.*

The proof of Proposition 3.17 (respectively Proposition 3.19) is similar to the proof of Proposition 3.9 (respectively Proposition 3.11).

The composition is the counterpart for (Comp-D-DPAS) of the addition for (Add-D-DPAS) (function a).

Definition 3.20 (Circuit U_c)**.** *We note the antecedent set* $S_c = 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T \times E \times 2^{[n]} \times (\mathbb{Z} \cup \{+\infty\}) \times T \times E$*. Let* $c : S_c \to \mathbb{Z} \cup \{+\infty\}$ *be the function:*

$$c(\omega_1^s, \omega_1^v, \omega_1^t, \omega_1^e, \omega_2^s, \omega_2^v, \omega_2^t, \omega_2^e) = \begin{cases} \omega_1^v & \text{if } \omega_1^t = \omega_2^v \\ +\infty & \text{else} \end{cases}$$

We note U_c *the quantum circuit corresponding to c, namely:*

$$\forall (\omega_1^s, \omega_1^v, \omega_1^t, \omega_1^e, \omega_2^s, \omega_2^v, \omega_2^t, \omega_2^e) \in S_c, \quad U_c \left|\omega\right\rangle \left|0\right\rangle = \left|\omega\right\rangle \left|c(\omega)\right\rangle \ ,$$

where $\left|\omega\right\rangle = \left|\omega_1^s\right\rangle \left|\omega_1^v\right\rangle \left|\omega_1^t\right\rangle \left|\omega_1^e\right\rangle \left|\omega_2^s\right\rangle \left|\omega_2^v\right\rangle \left|\omega_2^t\right\rangle \left|\omega_2^e\right\rangle$ *is encoded in eight registers.*

Notice that the function c is meant to be applied on objects of $\Omega_{\text{comp}}(J, t, \epsilon)$, for $J \subseteq [n]$, $t \in T$ and $\epsilon \in E$, according to recurrence (Comp-D-DPAS).

Proposition 3.21 (Complexity of U_c)**.** *The complexity of* U_c *is polynomial in the size of the input.*

Proof. This is the same proof as for Proposition 3.13. □

Remark 3.22. *Notice that, for* $J \subseteq [n]$*,* $t \in T$ *and* $\epsilon \in E$*,*

$$\text{OPT}_\epsilon[J, t] = \min_{\omega \in \Omega_{comp}(J,t,\epsilon)} c(\omega)\,. \tag{7}$$

3.2 Description of the algorithm for Additive DPAS

In this section, we describe our hybrid algorithm Q-DDPAS adapted from the work of Ambainis et al. [3] in the gate-based quantum computing model. 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 auxiliary problem P' satisfies recurrences (Comp-DPAS) and (Comp-D-DPAS) derives directly, as we explain later in Subsection 3.3. 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 problem P on sub-instances of size $n/4$. Second, we call recursively two times QMF on Equation (Add-D-DPAS) to find optimal values

of problem P on sub-instances of size $n/2$ and eventually of size n (corresponding to the nominal problem $\mathcal{P}$).

Before presenting our hybrid algorithm, we introduce some notations about indexing quantum circuits to be able to describe rigorously the quantum circuits of Q-DDPAS. Let $reg = |q_1\rangle \dots |q_n\rangle$ be a register of n qubits and U be an operator acting on k qubits, with $k < n$. Let I be a k-tuple of distinct indices in $[n]$, $I = (i_1, \dots, i_k)$. We denote by U^I the operator acting on the full register reg, that applies U on $|q_{i_1}\rangle \dots |q_{i_k}\rangle$, and applies Id on the remaining qubits. For instance, if I is the tuple of contiguous indices $(3, \dots, k+3)$ with $k < n-3$, then

$$U^I := Id^{\otimes 2} \otimes U \otimes Id^{\otimes n-k-3}\,.$$

For $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_l)$ two distinct tuples in $[n]$ (k-tuple and l-tuple where $i \neq j, \forall (i,j) \in I \times J$) , we note $I \oplus J$ the concatenation of I and J, namely $I \oplus J = (i_1, \dots, i_k, j_1, \dots, j_l)$. Regarding the QMF operator, let us denote the indexes related to the quantum circuit U_f of a function f as

$$U_f \underbrace{|i\rangle}_{I} \underbrace{|0\rangle}_{J} = \underbrace{|i\rangle}_{I} \underbrace{|f(i)\rangle}_{J}.$$

To clarify the computations detailed next, we index the corresponding QMF operator as $U_{\mathrm{QMF}}[U_f^I]$. We omit the index J because this is an *auxiliary* register that does not appear in the output of $U_{\mathrm{QMF}}[U_f]$.

We present the quantum circuits used in the quantum part, as well as the numbering of the different registers.

- Let $|\mathrm{ini}\rangle$ be the initial state:

$$|\mathrm{ini}\rangle := \underbrace{|[n]\rangle\,|0\rangle}_{I^1} \underbrace{|0\rangle^{\otimes 3}}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|0\rangle^{\otimes 3}}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

  where the tuples indexing the different registers are decomposed as follows:

$$I^1 = I_1^1 \oplus I_2^1$$
$$I^2 = I_1^2 \oplus I_2^2 \oplus I_3^2$$
$$I^3 = I_1^3 \oplus I_2^3$$

$$I^4 = I_1^4 \oplus I_2^4 \oplus I_3^4$$

$$I^5 = I_1^5 \oplus I_2^5$$

$$I^6 = I_1^6 \oplus I_2^6$$

- Let

$$U_{\text{ini}} := (U_{\Omega_{\text{add}}}^{I^2} \otimes U_{\Omega_{\text{add}}}^{I^4}) \cdot U_{\Lambda_{\text{add}}}^{I^1 \oplus I^2 \oplus I^4} \tag{8}$$

  be the quantum circuit that, given initial quantum state $|\text{ini}\rangle$, superposes all the couples (X, X') such that $X, X' \subseteq [n]$, $|X| = |X'| = n/4$ and $X \cap X' = \emptyset$. For each couple, the optimal values and parameters associated are also superposed.

- The quantum circuit $U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2}] \otimes U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4}]$ applies two QMF *in parallel* (resulting from the tensor product of two quantum circuits) on the function a. Consequently, let

$$U_{\text{recur1}} := U_a^{I_1^2 \oplus I_2^3 \oplus I_1^4 \oplus I_2^5 \oplus I_2^1} \left(U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2}] \otimes U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4}] \right)$$

  be the quantum circuit that adds, with the help of of function a, the resulting values of the two registers.

- Eventually, let

$$U_{\text{recur}} := U_{\text{QMF}}^{I_1^2 \oplus I_2^3 \oplus I_1^4 \oplus I_2^5 \oplus I_2^1 \oplus I^6}[U_{\text{recur1}}] \tag{9}$$

  be the quantum circuit that applies QMF on the function represented by the circuit U_{recur1}.

The bounded-error hybrid algorithm Q-DDPAS is described in Algorithm 1.

Theorem 3.23. *The bounded-error algorithm Q-DDPAS (Algorithm 1) solves $\mathcal{P}$ in $\mathcal{O}^*(|T| \cdot 1.754^n)$.*

The proof of Theorem 3.23 relies on the two lemmas introduced next. However, before stating and proving these lemmas, we observe that the complexity of Q-DDPAS can be further reduced by performing a third call to Equation (Add-D-DPAS) as suggested in [3].

Observation 3.24. *A slight modification of Q-DDPAS reduces the complexity to $\mathcal{O}^*(|T| \cdot 1.728^n)$.*

For the sake of clarity, we will prove Observation 3.24 only after having proved Theorem 3.23. We now introduce the two lemmas necessary to prove Theorem 3.23.

Input: Problem P satisfying (Add-DPAS) and (Add-D-DPAS)
Output: OPT$[[n], 0]$ with high probability
begin classical part
  for $X \subseteq [n] : |X| = n/4$ *and* $t \in T$ **do**
    Compute the optimal value OPT$[X, t]$ and the corresponding permutation $\pi^*[X, t]$ by classical (Add-DPAS);
    Store the tuple $(X, t, \text{OPT}[X, t], \pi^*[X, t])$ in the QRAM;
  end
end
begin quantum part
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}}U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes I_2^6;
end
Return the outcome of the measurement

Algorithm 1: Q-DDPAS for Additive DPAS

Lemma 3.25. *The optimal value of $\mathcal{P}$ is stored in the register of indexes I_2^6 by Q-DDPAS with high probability.*

Proof. We provide next a sketch of the proof, referring to Appendix B for the details of the computations. We give some intuition on the effect of the quantum circuit $U_{\text{recur}}U_{\text{ini}}$ and start by explaining the effect of U_{ini} defined in (8). First, the application of $U_{\Lambda_{\text{add}}}$ superposes all elements of $\Lambda_{\text{add}}([n], 0)$ in the registers of indexes I^2 (partition of J) and I^4 (partition of $[n] \setminus J$). This essentially amounts to superposing all the $\binom{n}{n/2}$ bi-partitions of $[n]$ where each partition is of size $n/2$ (parameters t included). Next, we apply $U_{\Omega_{\text{add}}}$ on register of index I^2, respectively I^4. This superposes all elements of $\Omega_{\text{add}}(J, t)$ (for a J of size $n/2$ and $t \in T$ previously described in registers of indexes I^2, respectively I^4). This essentially amounts to superposing all the $\binom{n/2}{n/4}$ bi-partitions of $[n]$ where each partition is of size $n/2$, parameters t included, and the optimal value associated already stored in the QRAM.

Let us explain the effect of U_{recur} defined in (9). The application of $U_{\text{QMF}}[U_a]$ on a register encoding (J, t) and the superposition of elements of $\Omega_{\text{add}}(J, t)$ stores OPT$[J, t]$ (with high probability) in an output register, according to Equation (Add-D-DPAS). Thus, $U_{\text{QMF}}[U_a]$ on register of index I^2, respectively I^4, superposes all OPT$[J, t]$ in I^3, respectively I^5, according to Remark 3.14. In other words, the circuit $U_{\text{QMF}}^{I_3^2 \oplus I^3}[U_a^{I_3^2 \oplus I_1^3}] \otimes (U_{\text{QMF}}^{I_3^4 \oplus I^5}[U_a^{I_3^4 \oplus I_1^5}]$ that appears in U_{recur1} superposes (with high probability) all optimal values of Equation (Add-D-DPAS) for J of size $n/2$. Now that the

optimal values are known for sets of size $n/2$ (before, we only knew optimal values for sets of size $n/4$), we apply one more time $U_{\mathrm{QMF}}[U_a]$ on these new registers: it outputs $\mathrm{OPT}[[n], 0]$ with high probability on the register of index I_2^6. □

Lemma 3.26. *The complexity of Q-DDPAS is* $\mathcal{O}^*\left(|T| \cdot 1.754^n\right)$.

Proof. Let us compute the complexity of this algorithm. First, we compute the complexity of the classical part. The proof of Lemma 2.2 shows that solving all $\mathrm{OPT}[X, t]$ for all X of size $n/4$ and for all $t \in T$ is done by (Add-DPAS) in time

$$|T| poly(n) \sum_{k=1}^{n/4} k \binom{n}{k} = \mathcal{O}^*\left(|T| \binom{n}{\leq n/4}\right).$$

Thus, because $\mathcal{O}^*\left(\binom{n}{\leq n/4}\right) = \mathcal{O}^*(2^{0.811n})$ (see Equation (19)), the complexity of the classical part is

$$\mathcal{O}^*(|T| \cdot 2^{0.811n}).$$

Second, let us compute the complexity of the quantum part (using Property 3.3).

- The complexity of U_{ini} is polynomial in n. Indeed, $U_{\Lambda_{\mathrm{add}}}$ is polynomial in n (Property 3.9). Moreover, $U_{\Omega_{\mathrm{add}}}$ is also polynomial in n: the classical part stored in the QRAM all $\mathrm{OPT}[X, t]$ for X of size $n/4$ and $t \in T$ (Property 3.11).
- The complexity of U_{recur} is $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}\right)$. Indeed, both terms $U_{\mathrm{QMF}}[U_a]$ in U_{recur1} have a polynomial complexity for U_a and find the minimum of functions with a domain of size $\binom{n/2}{n/4}$. Thus, the complexity of each of these two factors is $\mathcal{O}^*\left(\sqrt{\binom{n/2}{n/4}}\right)$, and so is the complexity of the tensor product. The circuit U_{recur1} has the same complexity because of the composition with U_a that is polynomial. The circuit U_{recur} finds the minimum of a function with a domain of size $\binom{n}{n/2}$ described by the corresponding quantum circuit U_{recur1} above. Thus, its complexity is $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}\right)$.

Because $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}\right) = \mathcal{O}^*(2^{0.75n})$ (see Equation (21)), the complexity of the quantum part is

$$\mathcal{O}^*(2^{0.75n}).$$

Eventually, the complexity of Q-DDPAS is

$$\mathcal{O}^*\left(2^{0.75n} + |T| \cdot 2^{0.811n}\right) = \mathcal{O}^*\left(|T| \cdot 2^{0.811n}\right) = \mathcal{O}^*\left(|T| \cdot 1.754^n\right) .$$

□

Proof of Theorem 3.23. Follows directly from Lemmas 3.25 and 3.26. □

Proof of Observation 3.24. The slight modification of Q-DDPAS amounts to adding a level of recurrence in the quantum part, but instead of searching for the best concatenation among all the bi-partition of size $(n/8, n/8)$ (i.e. solving Equation (Add-D-DPAS) for $|J| = n/4$), we search for the best concatenation among all the bi-partitions of size $(0.945 \cdot \frac{n}{4}, 0.055 \cdot \frac{n}{4})$, i.e. solving

$$\mathrm{OPT}[J,t] = \min_{\substack{X \subseteq J \\ |X| = 0.945|J|}} \left\{ \mathrm{OPT}[X,t] + h(J,X,t) + \mathrm{OPT}[J \setminus X, t_{\mathrm{shift}}(J,X,t)] \right\}.$$

A third call to this recurrence formula in Q-DDPAS implies that:

- the classical part computes $\mathrm{OPT}[X,t]$ for X of size $0.945 \cdot \frac{n}{4}$ and $0.055 \cdot \frac{n}{4}$. Its complexity is then $\mathcal{O}^*\left(|T| \binom{n}{\leq 0.945 \cdot \frac{n}{4}}\right) = \mathcal{O}^*(|T| \cdot 2^{0.789n})$ (see Equation (20)).
- the quantum part applies three levels of recurrence of QMF, finding the minimum over functions with a domain of size $\binom{n}{n/2}$, $\binom{n/2}{n/4}$ and $\binom{n/4}{0.945 \cdot n/4}$ respectively. Its complexity is then $\mathcal{O}^*\left(\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}\binom{n/4}{0.945 \cdot n/4}}\right) = \mathcal{O}^*(2^{0.789n})$ (see Equation (22)).

The quantum part and the classical part have the same complexity, thus the total complexity of Q-DDPAS is the same, namely $\mathcal{O}^*(2^{0.789n}) = \mathcal{O}^*(1.728^n)$. □

Notice that the classical part of Q-DDPAS can be replaced by any classical algorithm $\mathcal{A}$, if $\mathcal{A}$ computes in $\mathcal{O}^*(|T| \cdot 1.728^n)$ all $\mathrm{OPT}[X,t]$ for $X \subseteq [n]$ of size $n/4$ and $t \in T$. Moreover, if $\mathcal{A}$ happens to reduce the classical part complexity $\mathcal{O}^*(|T| \cdot c^n)$ for $c < 1.728$, the complexity of Q-DDPAS can also be reduced in the same spirit as the slight modification of Observation 3.24.

We illustrate in Table 1 the worst-case time complexities of solving the three NP-hard scheduling problems examples introduced in Subsection 2.2 ($1|\tilde{d}_j| \sum_j w_j C_j$, $1|| \sum_j w_j T_j$ and $1|prec| \sum_j w_j C_j$) with Q-DDPAS and compare them with the complexities of the best-known current classical al-

gorithms. Q-DDPAS improves the complexity of the exponent but sometimes at the cost of a pseudo-polynomial factor ($\sum p_j$ for problems $1|\tilde{d_j}|\sum_j w_j C_j$ and $1||\sum_j w_j T_j$).

Problem	Q-DDPAS	Best classical algorithm
$1\|\tilde{d_j}\|\sum w_j C_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(2^n)$, T'kindt et al. [32]
$1\|\|\sum w_j T_j$	$\mathcal{O}^*\left(\sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(2^n)$, T'kindt et al. [32]
$1\|prec\|\sum w_j C_j$	$\mathcal{O}^*\left(1.728^n\right)$	$\mathcal{O}^*((2-\epsilon)^n)$, for small ϵ, Cygan et al. [6]

Table 2: Comparison of complexities between Q-DDPAS and the best-known classical algorithm for some scheduling problems satisfying (Add-DPAS) and (Add-D-DPAS)

3.3 Adaptation to Composed DPAS

In the previous subsection, we describe Q-DDPAS for a problem $\mathcal{P}$ for which its related problem P satisfies both recurrences (Add-DPAS) and (Add-D-DPAS). However, the description of Q-DDPAS for a problem $\mathcal{P}$ related to auxiliary problem P' satisfying recurrences (Comp-DPAS) and (Comp-D-DPAS) derives naturally. It essentially amounts to replacing Λ_{add} by Λ_{comp}, Ω_{add} by Ω_{comp} and function a by function c. Consequently, the quantum circuit $U_{\Lambda_{\text{comp}}}$, respectively $U_{\Omega_{\text{comp}}}$, apply on 8 registers, respectively 4 registers, that differ from Q-DDPAS for Additive DPAS. Moreover, Q-DDPAS does not solve directly $\mathcal{P}$ but the auxiliary problem $P'([n], 0, \epsilon_0)$, for $\epsilon_0 \in E$. Eventually, we use it as a subroutine in a meta-algorithm to solve $\mathcal{P}$.

Let us describe the slightly different quantum circuits adapting the number of registers and the registers on which they apply. Let $\epsilon_0 \in E$. The initial state is

$$|\text{ini}\rangle = \underbrace{|[n]\rangle\,|0\rangle\,|\epsilon_0\rangle}_{I^1}\underbrace{|0\rangle^{\otimes 4}}_{I^2}\underbrace{|0\rangle^{\otimes 2}}_{I^3}\underbrace{|0\rangle^{\otimes 4}}_{I^4}\underbrace{|0\rangle^{\otimes 2}}_{I^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

where the tuples indexing the different registers are decomposed as follows:

$$I^1 = I_1^1 \oplus I_2^1 \oplus I_3^1$$
$$I^2 = I_1^2 \oplus I_2^2 \oplus I_3^2 \oplus I_4^2$$
$$I^3 = I_1^3 \oplus I_2^3$$
$$I^4 = I_1^4 \oplus I_2^4 \oplus I_3^4 \oplus I_4^4$$

$$I^5 = I_1^5 \oplus I_2^5$$

$$I^6 = I_1^6 \oplus I_2^6$$

The three quantum circuits that appear on the quantum part are:

$$U_{\text{ini}} = (U_{\Omega_{\text{comp}}}^{I^2} \otimes U_{\Omega_{\text{comp}}}^{I^4}) \cdot U_{\Lambda_{\text{comp}}}^{I^1 \oplus I^2 \oplus I^4},$$

$$U_{\text{recur1}} = U_c^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4} \left(U_{\text{QMF}}^{I_4^2 \oplus I^3}[U_c^{I_4^2}] \otimes U_{\text{QMF}}^{I_4^4 \oplus I^5}[U_c^{I_4^4}] \right),$$

$$U_{\text{recur}} = U_{\text{QMF}}^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4 \oplus I^6}[U_{\text{recur1}}].$$

The adaptation of Q-DDPAS to solve $P'([n], 0, \epsilon_0)$ for a given $\epsilon_0 \in E$ is described in Algorithm 2.

Input: $\epsilon_0 \in E$, auxiliary problem P' satisfying (Comp-DPAS) and (Comp-D-DPAS)
Output: $\text{OPT}[[n], 0, \epsilon_0]$ with high probability
begin classical part
  for $X \subseteq [n] : |X| = n/4$ *and* $t \in T$ **do**
    Compute the optimal value $\text{OPT}[X, t, \epsilon_0]$ and the corresponding permutation $\pi^*[X, t, \epsilon_0]$ by classical (Comp-DPAS);
    Store the tuple $(X, t, \text{OPT}[X, t, \epsilon_0], \pi^*[X, t, \epsilon_0])$ in the QRAM;
  end
end
begin quantum part
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}} U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes I_2^6;
end
Return the outcome of the measurement

Algorithm 2: Q-DDPAS for Composed DPAS

Theorem 3.27. *Let $\epsilon_0 \in E$. The bounded-error algorithm Q-DDPAS (Algorithm 2) solves $P'([n], 0, \epsilon_0)$ in $\mathcal{O}^*(|E|^2 \cdot |T| \cdot 1.754^n)$.*

Proof. First, the optimal value of $P'([n], 0, \epsilon_0)$ is stored in the register of indexes I_2^6 with high-probability. This is exactly the same reasoning as the proof for Q-DDPAS for Additive DPAS. The classical part computes and stores all $\binom{n/2}{n/4}$ bi-partitions of $[n]$, for any parameter in T, any value

in E and the optimal value associated in the QRAM. The quantum part applies recursively two times QMF. The first call put in superposition, thanks to Remark 3.22, the optimal values for sets of size n/2. The second call finds the optimal value for $[n]$.

Second, the computation of the complexity is similar. The main changes are that we require to store values in E in Ω_{comp}, but also let vary parameters in T to implement the composition function c. This gives an extra factor $|E| \cdot |T|$ in the quantum part. We give the general idea of the computation of complexities for each part:

- Classical part: according to Lemma 2.8, (Comp-DPAS) computes all $\text{OPT}[X, t, \epsilon]$ for X of size n/4, $t \in T$ and $\epsilon \in E$, in

$$|E|^2 \cdot |T| poly(n) \sum_{k=1}^{n/4} k \binom{n}{k} = \mathcal{O}^* \left(|E|^2 \cdot |T| \binom{n}{\leq n/4} \right) = \mathcal{O}^*(|E|^2 \cdot |T| \cdot 2^{0.811n}) .$$

- Quantum part: the first call to QMF *in parallel* is done on a set of size $|E| \cdot |T| \cdot \binom{n/2}{n/4}$. The second call to QMF is done on a set of size $|E| \cdot |T| \cdot \binom{n}{n/2}$. Eventually, the complexity of the quantum part is:

$$\begin{aligned} \mathcal{O}^* \left(\sqrt{|E| \cdot |T| \cdot \binom{n/2}{n/4}} \sqrt{|E| \cdot |T| \cdot \binom{n}{n/2}} \right) &= \mathcal{O}^* \left(|E| \cdot |T| \sqrt{\binom{n/2}{n/4} \binom{n}{n/2}} \right) \\ &= \mathcal{O}^*(|E| \cdot |T| \cdot 2^{0.75n}) . \end{aligned}$$

Eventually, the total complexity is the maximum of the classical and the quantum part, namely,

$$\mathcal{O}^*(|E|^2 \cdot |T| \cdot 2^{0.811n}) = \mathcal{O}^* \left(|E|^2 \cdot |T| \cdot 1.754^n \right) .$$

□

We naturally obtain the hybrid Algorithm 3 that takes Q-DDPAS as a subroutine.

Corollary 3.28. *The bounded-error Algorithm 3, with Q-DDPAS as a subroutine, solves $\mathcal{P}$ in $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 1.754^n)$.*

Proof. Algorithm 3 calls $|E|$ times Q-DDPAS for Composed DDPAS (Algorithm 2), which complexity is given in Theorem 3.27. □

Input: Auxiliary problem P' satisfying (Comp-DPAS) and (Comp-D-DPAS)
Output: $\min_{\epsilon \in E}\left\{\epsilon : \mathrm{OPT}[[n], 0, \epsilon] < +\infty\right\}$ with high probability
$\epsilon^* \leftarrow +\infty$;
for $\epsilon \in E$ **do**
  Solve $P([n], 0, \epsilon)$ with Algorithm 2;
  if $\mathrm{OPT}[[n], 0, \epsilon] < +\infty$ *and* $\epsilon < \epsilon^*$ **then**
    $\epsilon^* \leftarrow \epsilon$;
  end
end
Return ϵ^*

Algorithm 3: Meta-algorithm with subroutine Q-DDPAS for Composed DPAS

As for the case of Q-DDPAS for Additive DPAS, we can reduce the exponential part of Q-DDPAS complexity for Composed DPAS, by the very same modification.

Observation 3.29. *A slight modification of the Q-DDPAS algorithm can reduce the complexity of Algorithm 3 to* $\mathcal{O}^*(|E|^3 \cdot |T| \cdot 1.728^n)$.

Proof. Refer to the proof of Observation 3.24. □

We synthesize in Table 3 the worst-case time complexity achieved by Q-DDPAS on the examples of scheduling problems satisfying (Comp-DPAS) and (Comp-D-DPAS) and compare it with the best-known classical complexity. The latter comes from the algorithm of Inclusion-Exclusion designed by Ploton and T'kindt [25], which provides a generic method to solve such problems. We observe that Q-DDPAS improves the exponential part of the complexity, at a cost of a higher degree for the pseudo-polynomial factor.

Problem	Q-DDPAS	Best classical algorithm
$1\|r_j\|\sum w_jU_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot \sum p_j \cdot 1.728^n\right)$	$\mathcal{O}^*(\sum w_j \cdot \sum p_j \cdot 2^n)$, Ploton and T'kindt [25]
$1\|r_j\|\sum w_jC_j$	$\mathcal{O}^*\left((\sum w_j)^3 \cdot (\sum p_j)^4 \cdot 1.728^n\right)$	$\mathcal{O}^*(\sum w_j \cdot (\sum p_j)^2 \cdot 2^n)$, Ploton and T'kindt [25]

Table 3: Comparison of complexities between Q-DDPAS and the best-known classical algorithm for some scheduling problems satisfying (Comp-DPAS) and (Comp-D-DPAS)

4 Adaptation to decision problems

We saw in the previous section that the recurrence to solve $\mathcal{P}$ can be done on a minimization problem, possibly involving an auxiliary problem. Sometimes, the recurrence does not apply directly to a minimization problem but to a *decision* problem. In this section, we adapt the hybrid algorithm Q-DDPAS to this case. Originally, this adaptation came from the desire to solve the 3-machine flowshop problem. Because this is the only scheduling problem we found that applies to this resolution, henceforth we describe the dynamic programming properties for the 3-machine flowshop problem to avoid additional and unnecessary abstraction. Notice that it can still be generalized to other problems with the same structure. Particularly, it easily generalizes to the m-flowshop problem, for $m \geq 4$.

4.1 3-machine flowshop problem and dynamic programming

We consider the permutation flowshop problem on 3 machines for n jobs with minimizing the makespan as the objective function. This strongly NP-hard problem is often referred to as $F3||C_{\max}$ in the literature, as mentioned by Shang et al. [28]. Each job $j \in [n]$ consists of 3 operations O_{ij} for $i \in [3]$, each operation being processed on the i-th machine. We note p_{ij} the processing time of operation O_{ij}. Each machine performs at most one operation at a time. For each job j, operations must be processed in the specific order O_{1j}, O_{2j}, O_{3j}: the first operation gets processed on the first machine, then the second operation gets processed on second machine (as soon as the first operation is finished and the machine 2 is available), and eventually the third operation gets processed on the third machine (as soon as the second operation is finished and the machine 3 is available). Thus, only the order of the executions of jobs can be decided. It implies that a solution is entirely described by the permutation of jobs on the first machine, so the problem can be formulated as

$$\min_{\pi \in S_{[n]}} C_{\max}(\pi)\,, \tag{10}$$

where $C_{\max}$ is the maximum completion time.

It happens that the two techniques presented so far do not apply to (10) so we present in the next section an alternative approach involving the decision counterpart of the above optimization problem.

4.1.1 Decision problems definitions

Let us define the decision problem D related to $\mathcal{P}$. An instance $\mathcal{J}$ of D is described by instance $\mathcal{I}$ of $\mathcal{P}$ together with an additional parameter $t \in \mathbb{Z}$. As in Subsection 2.1, we consider the sub-instance $\mathcal{J}(J) = (\mathcal{I}(J), t)$ corresponding to $J \subseteq [n]$ and we focus on the solution of

$$D(J,t): \qquad \bigvee_{\pi \in \Pi(J,t)} f_{\text{bool}}(\pi, J, t)\,, \tag{11}$$

where $\Pi(J,t) \subseteq S_J$ is the set of feasible permutations of J and $f_{\text{bool}}(.,J,t)$ is a boolean function. We note $D[J,t]$ the boolean value of $D(J,t)$. As before, we introduce the bounded set $T(\mathcal{I}) \subseteq \mathbb{Z}$ and omit the dependency to $\mathcal{I}$ and note T this latter set.

We assume that D and $\mathcal{P}$ are related through:

$$\mathcal{P}: \qquad \min_{t \in T} \Big\{ t : D[[n],t] = 1 \Big\}\,.$$

4.1.2 Special case of 3-machine flowshop

Let us describe D for the special case of the 3-machine flowshop at hand. We slightly modify the description of an instance, and assume that an instance $\mathcal{J}$ of D is an instance $\mathcal{I} = ([n], p_{ij} : i \in [3], j \in [n])$ of $\mathcal{P}$ together with four parameters $\beta_2, \beta_3, \epsilon_2, \epsilon_3 \in \mathbb{Z}$. We introduce the bounded set

$$T(\mathcal{I}) = \left\{ 0, \ldots, \sum_{j \in [n], i \in [3]} p_{ij} \right\} \subseteq \mathbb{Z}$$

and consider parameters $(\beta_2, \beta_3, \epsilon_2, \epsilon_3) \in T(\mathcal{I})^4$. Henceforth, we denote by T the set $T(\mathcal{I})$ to lighten the notations. Notice that the number of parameters is four for the 3-machine flowshop, but generalizes to $2(m-1)$ parameters for the m-machine flowshop as we see later.

Definition 4.1 (Decision problem)**.** *For $J \subseteq [n]$, $\vec{\beta} = (\beta_2, \beta_3) \in T^2$ and $\vec{\epsilon} = (\epsilon_2, \epsilon_3) \in T^2$, we define the decision problem $D(J, \vec{\beta}, \vec{\epsilon})$ on a sub-instance associated with J as the following question:*

> *"Does there exist a permutation $\pi \in S_J$ such that, for $i \in \{2,3\}$, $b_i(\pi) \geq \beta_i$, and $e_i(\pi) \leq \epsilon_i$?",*

where $b_i(\pi)$, respectively $e_i(\pi)$, denotes the time that the first operation begins, respectively the last operation ends, on the i-th machine, for $i \in [3]$.

In other words, problem $D(J, \vec{\beta}, \vec{\epsilon})$ is asking whether there exists a feasible permutation with jobs in J such that it *holds* between the two *temporal fronts* $\vec{\beta}$ and $\vec{\epsilon}$. Notice that it is not necessary to impose any beginning and ending time for the first machine ($i = 1$). Indeed, the problem is time-invariant, thus we can always consider that the scheduling problem starts at time 0, and that the total completion time on the first machine is known and equal to the sum of processing times of considered jobs. With these notations, $\mathcal{P}$ can be cast as follows:

$$\mathcal{P}: \qquad \min_{c \in T} \Big\{ c : D[[n], (0, c), (0, c)] = 1 \Big\}. \tag{12}$$

The decision problem D introduced in Definition 4.1 satisfies the recurrence (Dec-DPAS) below.

Property 4.2 (Decision DPAS)**.** *For all $J \subseteq [n]$ of even cardinality, $\vec{\beta} \in T^2$ and $\vec{\epsilon} \in T^2$,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\substack{X \subseteq J : |X| = |J|/2, \\ \vec{t} \in [\vec{\beta}, \vec{\epsilon}]}} \Big(D[\{j\}, \vec{\beta}, \vec{t}] \wedge D[J \setminus \{j\}, \vec{t} \ominus p_{1j}, \vec{\epsilon} \ominus p_{1j}] \Big) , \tag{Dec-DPAS}$$

where $\vec{t} \in [\vec{\beta}, \vec{\epsilon}]$ means that the i-th coordinate of $\vec{t}$ is between the i-th coordinates of $\vec{\beta}$ and $\vec{\epsilon}$, and where the operation $\vec{v} \ominus c$, for a vector $\vec{v}$ and a constant c, subtracts c to each coordinate of $\vec{v}$.

This latter recurrence enables $\mathcal{P}$ to be solved by classical dynamic programming.

Lemma 4.3. (Dec-DPAS) *solves $\mathcal{P}$ in $\mathcal{O}^*(|T|^4 \cdot 2^n)$.*

Proof. First, we can show that, for a given $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$, (Dec-DPAS) solves $D([n], \vec{\beta}_0, \vec{\epsilon}_0)$ in $\mathcal{O}^*(|T|^4 \cdot 2^n)$. This is essentially the same lines of the proof for Lemma 2.2. Second, to solve $\mathcal{P}$, we make a dichotomic search over T to find the minimum $c \in T$ such that $D([n], (0, c), (0, c))$ is true according to Equation 12. Thus, (Dec-DPAS) is called $\log_2(|T|)$ times. Because $|T| = \sum p_{ij}$ is a pseudo-polynomial term of the instance, the total complexity is

$$\mathcal{O}^*(\log_2(|T|) \cdot |T|^4 \cdot 2^n) = \mathcal{O}^*(|T|^4 \cdot 2^n).$$

□

Not only does problem D satisfy (Dec-DPAS), but it also satisfies the following (Dec-D-DPAS) recurrence.

Property 4.4 (Decision Dichotomic DPAS)**.** *For all $J \subseteq [n]$ of even cardinality, $\vec{\beta} \in T^2$ and $\vec{\epsilon} \in T^2$,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\substack{X \subseteq J : |X| = |J|/2, \\ \vec{t} \in [\vec{\beta}, \vec{\epsilon}]}} \left(D[X, \vec{\beta}, \vec{t}] \wedge D[J \setminus X, \vec{t} \ominus \sum_{j \in X} p_{1j}, \vec{\epsilon} \ominus \sum_{j \in X} p_{1j}] \right) . \qquad \text{(Dec-D-DPAS)}$$

Lemma 4.5. (Dec-D-DPAS) *solves $\mathcal{P}$ in $\omega(|T|^4 \cdot 2^n)$.*

Proof. This proof is similar to the proof of Lemma 2.5, with the argument that dichotomic search is polynomial in the size of the instance as in the proof of Proposition 4.3. □

Once again, we observe that recurrence (Dec-DPAS) outperforms recurrence (Dec-D-DPAS) to solve by classical dynamic programming our problem $\mathcal{P}$. In the next section, we describe how we adapt Q-DDPAS to take advantage of those two recurrences to solve the 3-machine flowshop problem.

4.2 Hybrid algorithm Q-Dec-DDPAS

We call Q-Dec-DDPAS the adapted decision version of Q-DDPAS. The main difference is that instead of searching for a minimum value in a set in recurrence (Add-D-DPAS) or (Comp-D-DPAS), we search for a True value in a set in recurrence (Dec-D-DPAS). Thus, it essentially amounts to replacing QMF with the algorithm of Grover Search specified below.

Definition 4.6 (Circuit U_G)**.** *Let $f : [n] \to \{0, 1\}$ be a function and let U_f be its corresponding quantum circuit, specifically,*

$$U_f \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left|f(i)\right\rangle , \quad \forall i \in [n] .$$

We note $U_G[U_f]$ the quantum circuit corresponding to the algorithm of Grover [12] that computes with high probability the logical OR of all the f values. If it appends to be True, $U_G[U_f]$ also gives the corresponding set $I_f = \{i : f(i) = 1\}$. Specifically,

$$U_G[U_f] \sum_{i=1}^{N} \frac{1}{\sqrt{N}} \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle = \sum_{i=1}^{N} \frac{1}{\sqrt{N}} \left|i\right\rangle \left|I_f\right\rangle \left| \bigvee_{i \in [N]} f(i) \right\rangle ,$$

Observation 4.7 (Complexity of U_G)**.** *The complexity of Grover Search is* $\mathcal{O}\left(\sqrt{n}\cdot C_f(n)\right)$*, where* n *is the size of the domain of* f *and* $\mathcal{O}(C_f(n))$ *is the complexity of the circuit* U_f*. Thus, according to Observation 3.3, the complexity of* $U_G[U_f]$ *is,*

$$\mathcal{O}\left(\sqrt{n}\cdot C_f(n)\right) .$$

In what follows, we define the sets and their associated quantum circuits to describe the Q-Dec-DDPAS algorithm.

Definition 4.8 (Sets Λ_{dec} and Ω_{dec})**.** *For* $J \subseteq [n]$ *such that* $|J|$ *is even and for* $\vec{\beta}, \vec{\epsilon} \in T^2$*, we define the set*

$$\Lambda_{dec}(J,\vec{\beta},\vec{\epsilon}) = \left\{ (X,\vec{\beta},\vec{t},J\setminus X,\vec{t},\vec{\epsilon}) : X \subseteq J, |X| = \frac{|J|}{2}, \vec{t} \in [\vec{\beta},\vec{\epsilon}] \right\} ,$$

and the set

$$\Omega_{dec}(J,\vec{\beta},\vec{\epsilon}) = \left\{ (X, D[X,\vec{\beta},\vec{t}],\vec{\beta},\vec{t},J\setminus X, D[J\setminus X,\vec{t},\vec{\epsilon}],\vec{t},\vec{\epsilon}) : X \subseteq J, |X| = \frac{|J|}{2}, \vec{t} \in [\vec{\beta},\vec{\epsilon}] \right\} .$$

The quantum circuits associated with these two sets are the following.

Definition 4.9 (Circuit $U_{\Lambda_{\text{dec}}}$)**.** *For* $J \subseteq [n]$ *such that* $|J|$ *is even, and for* $\vec{\beta}, \vec{\epsilon} \in T^2$*, we define* $U_{\Lambda_{dec}}$ *as follows:*

$$U_{\Lambda_{dec}} |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle |0\rangle^{\otimes 8} =$$
$$|J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle \sum_{(\lambda_1^s,\lambda_1^{tb},\lambda_1^{te},\lambda_2^s,\lambda_2^{tb},\lambda_2^{te})\in\Lambda_{dec}(J,\vec{\beta},\vec{\epsilon})} \frac{1}{\sqrt{|\Lambda_{dec}(J,\vec{\beta},\vec{\epsilon})|}} |\lambda_1^s\rangle \left|\lambda_1^{tb}\right\rangle |\lambda_1^{te}\rangle |0\rangle |\lambda_2^s\rangle \left|\lambda_2^{tb}\right\rangle |\lambda_2^{te}\rangle |0\rangle .$$

Notice that we index the objects that represent sets by s*, and the objects that represent scalars in* T^2 *by* tb *if it represents a couple of beginning times, or by* te *if it represents a couple of ending times.*

Proposition 4.10 (Complexity of $U_{\Lambda_{\text{dec}}}$)**.** *The complexity of* $U_{\Lambda_{dec}}$ *is polynomial in the size of the input.*

Definition 4.11 (Circuit $U_{\Omega_{\text{dec}}}$)**.** *For* $J \subseteq [n]$ *such that* $|J|$ *is even, and for* $\vec{\beta}, \vec{\epsilon} \in T^2$*, we define*

$U_{\Omega_{dec}}$ *as follows:*

$$U_{\Omega_{dec}} |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle |0\rangle = |J\rangle \left|\vec{\beta}\right\rangle |\vec{\epsilon}\rangle \sum_{\omega \in \Omega_{dec}(J,\vec{\beta},\vec{\epsilon})} \frac{1}{\sqrt{|\Omega_{add}(J,\vec{\beta},\vec{\epsilon})|}} |\omega\rangle \ .$$

Proposition 4.12 (Complexity of $U_{\Omega_{\text{dec}}}$)**.** *Let J be the input set. If we suppose to have stored in the QRAM the values $D[X,\vec{\beta},\vec{\epsilon}]$ for all $X \subseteq J$ such that $|X| = |J|/2$ and for all $\vec{\beta},\vec{\epsilon} \in T^2$, the complexity of $U_{\Omega_{dec}}$ is polynomial in the size of the input.*

The proof of Proposition 4.10, respectively Proposition 4.12, is similar to the proof of Proposition 3.9, respectively Proposition 3.11. Notice that $\vec{t} \in [\vec{\beta},\vec{\epsilon}]$ can be replaced by $\vec{t} \in T^2$ in sets $U_{\Lambda_{\text{dec}}}$ and $U_{\Omega_{\text{dec}}}$ so that the circuits that superpose all elements of these sets are easier to conceive. Indeed, (Dec-DPAS) and (Dec-D-DPAS) are less accurate but still valid with this replacement.

The operation in recurrence (Dec-D-DPAS) is not the addition (represented by the function a for (Add-D-DPAS)) nor the composition (represented by the function c for (Comp-D-DPAS)) but the logical AND. We define below its corresponding quantum circuit.

Definition 4.13 (Circuit U_{and})**.** *We note the antecedent set $S_{and} = 2^{[n]} \times \{0,1\} \times T^2 \times T^2 \times 2^{[n]} \times \{0,1\} \times T^2 \times T^2$. Let $and : S_{and} \to \{0,1\}$ be the function:*

$$and(\omega_1^s,\omega_1^b,\omega_1^{tb},\omega_1^{te},\omega_2^s,\omega_2^b,\omega_2^{tb},\omega_2^{te}) = \begin{cases} 1 & \text{if } \omega_1^b = \omega_2^b \\ 0 & \text{else} \end{cases}$$

We note U_{and} the quantum circuit associated to the function, specifically,

$$\forall \omega = (\omega_1^s,\omega_1^b,\omega_1^{tb},\omega_1^{te},\omega_2^s,\omega_2^b,\omega_2^{tb},\omega_2^{te}) \in S_{and}, \quad U_{and} |\omega\rangle |0\rangle = |\omega\rangle |and(\omega)\rangle \ .$$

Notice that objects representing boolean values are indexed by b. Note that according to recurrence (Dec-D-DPAS), the function *and* applies on objects of sets $\Omega_{\text{dec}}(J,\vec{\beta},\vec{\epsilon})$ for $J \subseteq [n]$ and $\vec{\beta},\vec{\epsilon} \in T^2$.

Proposition 4.14 (Complexity of U_{and})**.** *The complexity of U_{and} is polynomial in the size of the input.*

The proof of the above proposition is the same as the one of Proposition 3.13.

Remark 4.15. *Notice that for $J \subseteq [n]$ and $\vec{\beta}, \vec{\epsilon} \in T^2$,*

$$D[J, \vec{\beta}, \vec{\epsilon}] = \bigvee_{\omega \in \Omega_{dec}(J, \vec{\beta}, \vec{\epsilon})} and(\omega)\,. \tag{13}$$

Q-Dec-DDPAS algorithm solves $D([n], \vec{\beta}_0, \vec{\epsilon}_0)$, for $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$. Eventually, Q-Dec-DDPAS is the subroutine of a meta-algorithm that solves the 3-machine flowshop problem $\mathcal{P}$. Let us begin with the description of different quantum circuits for the quantum part of Q-Dec-DDPAS. Let $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$. The initial state is

$$|\text{ini}\rangle = \underbrace{|[n]\rangle \left|\vec{\beta}_0\right\rangle |\vec{\epsilon}_0\rangle}_{I^1} \underbrace{|0\rangle^{\otimes 4}}_{I^2} \underbrace{|0\rangle^{\otimes 2}}_{I^3} \underbrace{|0\rangle^{\otimes 4}}_{I^4} \underbrace{|0\rangle^{\otimes 2}}_{I^5} \underbrace{|0\rangle^{\otimes 2}}_{I^6},$$

where the tuples indexing the different registers are decomposed as follows:

$$\begin{aligned}
I^1 &= I_1^1 \oplus I_2^1 \oplus I_3^1 \\
I^2 &= I_1^2 \oplus I_2^2 \oplus I_3^2 \oplus I_4^2 \\
I^3 &= I_1^3 \oplus I_2^3 \\
I^4 &= I_1^4 \oplus I_2^4 \oplus I_3^4 \oplus I_4^4 \\
I^5 &= I_1^5 \oplus I_2^5 \\
I^6 &= I_1^6 \oplus I_2^6
\end{aligned}$$

The three quantum circuits that appear on the quantum part are:

$$U_{\text{ini}} = (U_{\Omega_{\text{dec}}}^{I^2} \otimes U_{\Omega_{\text{dec}}}^{I^4}) \cdot U_{\Lambda_{\text{dec}}}^{I^1 \oplus I^2 \oplus I^4},$$

$$U_{\text{recur1}} = U_{and}^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4} \left(U_G^{I_4^2 \oplus I^3}[U_{and}^{I_4^2}] \otimes U_G^{I_4^4 \oplus I^5}[U_{and}^{I_4^4}] \right),$$

$$U_{\text{recur}} = U_G^{I_1^2 \oplus I_2^3 \oplus I_2^2 \oplus I_3^2 \oplus I_1^4 \oplus I_2^5 \oplus I_2^4 \oplus I_3^4 \oplus I_6}[U_{\text{recur1}}]\,.$$

The description of Q-Dec-DDPAS is in Algorithm 4.

Theorem 4.16. *Let $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$. The bounded-error algorithm Q-Dec-DDPAS (Algorithm 4) solves*

Input: $\vec{\beta}_0, \vec{\epsilon}_0 \in T^2$, decision problem D satisfying (Dec-DPAS) and (Dec-D-DPAS)
Output: $D[[n], \vec{\beta}_0, \vec{\epsilon}_0]$ with high probability
begin classical part
  for $X \subseteq [n] : |X| = n/4$ *and* $\vec{\beta}, \vec{\epsilon} \in T^2$ **do**
    Compute the optimal value $D[X, \vec{\beta}, \vec{\epsilon}]$ and the corresponding permutation $\pi^*[X, \vec{\beta}, \vec{\epsilon}]$ by classical (Dec-DPAS);
    Store the tuple $(X, \vec{\beta}, \vec{\epsilon}, D[X, \vec{\beta}, \vec{\epsilon}], \pi^*[X, \vec{\beta}, \vec{\epsilon}])$ in the QRAM;
  end
end
begin quantum part
  Prepare quantum state $|\text{ini}\rangle$;
  Apply the quantum circuit $U_{\text{recur}} U_{\text{ini}}$ to $|\text{ini}\rangle$;
  Measure register of indexes I_2^6;
end
Return the outcome of the measurement

Algorithm 4: Q-Dec-DDPAS for 3-machine flowshop

$D([n], \vec{\beta}_0, \vec{\epsilon}_0)$ *in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.754^n)$.

Proof. As for the proof of Theorem 3.27, we follow the same reasoning of the proof of Theorem 3.23. First, we can show that the boolean value $D[[n], \vec{\beta}_0, \vec{\epsilon}_0]$ is stored in the register of indexes I_2^6 with high-probability. Indeed, the classical part computes and stores in the QRAM the decision variables for all $\binom{n/2}{n/4}$ bi-partitions of $[n]$, for any couple of parameters in T^2. The quantum part applies recursively two times Grover Search. The first call puts the optimal values for sets of size n/2 in superposition (refer to Remark 4.15). The second call finds the optimal value for $[n]$.

The computation of the complexity in time of Q-Dec-DDPAS is also similar.

- Classical part: according to Lemma 4.3, (Dec-DPAS) computes all $D[X, \vec{\beta}, \vec{\epsilon}]$ for X of size n/4 and $\vec{\beta}, \vec{\epsilon} \in T^2$ in
$$\mathcal{O}^*\left(|T|^4 \cdot \binom{n}{\leq n/4}\right) = \mathcal{O}^*(|T|^4 \cdot 2^{0.811n}) .$$
- Quantum part: the first call to Grover Search *in parallel* is done on a set of size $|T|^2 \cdot \binom{n/2}{n/4}$. The second call to Grover Search is done on a set of size $|T|^2 \cdot \binom{n}{n/2}$. Eventually, the complexity of the quantum part is:
$$\mathcal{O}^*\left(\sqrt{|T|^2 \cdot \binom{n/2}{n/4}}\sqrt{|T|^2 \cdot \binom{n}{n/2}}\right) = \mathcal{O}^*\left(|T|^2\sqrt{\binom{n/2}{n/4}\binom{n}{n/2}}\right)$$
$$= \mathcal{O}^*(|T|^2 \cdot 2^{0.75n}) .$$

Eventually, the total complexity is

$$\mathcal{O}^*(|T|^4 \cdot 2^{0.811n} + |T|^2 \cdot 2^{0.75n}) = \mathcal{O}^*\left(|T|^4 \cdot 1.754^n\right) = \mathcal{O}^*\left((\sum p_{ij})^4 \cdot 1.754^n\right)$$

□

It stems from Q-Dec-DDPAS the Algorithm 5 that solves the 3-machine flowshop $\mathcal{P}$.

Input: 3-machine flowshop
Output: Minimum makespan with high probability
$c^* \leftarrow +\infty$;
for $c \in T$ **do**
  Solve $D([n], (0, c), (0, c))$ with Algorithm 4;
  if $D[[n], (0, c), (0, c)]$ *and* $c < c^*$ **then**
    $c^* \leftarrow c$;
  end
end
Return c^*

Algorithm 5: Meta-algorithm with subroutine Q-Dec-DDPAS for the 3-machine flowshop

Theorem 4.17. *Algorithm 5 solves the 3-machine flowshop in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.754^n)$ *with high probability.*

Once again, as mentioned in Observation 3.24, the complexity can be reduced thanks to a slight modification on the Q-Dec-DDPAS that constitutes the subroutine.

Observation 4.18. *A slight modification of Algorithm 5 reduces the complexity of solving the 3-machine flowshop in* $\mathcal{O}^*((\sum p_{ij})^4 \cdot 1.728^n)$ *with high probability.*

It is worth noting that the previous algorithm easily generalizes to the m-machine flowshop problem and runs in $\mathcal{O}^*((\sum p_{ij})^{2(m-1)} \cdot 1.728^n)$. Indeed, the only difference is the description of the *temporal front* that necessitates $2(m-1)$ parameters.

This new method improves the best-known classical algorithm that is in $\mathcal{O}^*(3^n)$ or in $\mathcal{O}^*(M \cdot 2^n)$ if there exists a constant M such that $p_{ij} \leq M$ for all $i \in [3], j \in [n]$ presented by Shang et al. [28] and by Ploton and T'kindt [26]. Hybrid quantum-classical bounded-error Algorithm 5 reduces the exponential part of the time complexity at the cost of a pseudo-polynomial factor. For most cases, this factor is negligible because the numerical values of 3-machine flowshop instances are small

compared to the exponential part value. However, we present in the next subsection a way to dispose of this factor with an approximation scheme.

4.3 Approximation scheme

We present an approximation scheme for the 3-machine flowshop problem that trades the pseudo-polynomial factor in the complexity of Q-Dec-DDPAS and the optimality of the algorithm for a polynomial factor in $\frac{1}{\epsilon}$ and an approximation factor of $(1+\epsilon)$. In other words, we provide the Algorithm 6 that finds a solution in time $\mathcal{O}^*\left(\frac{1}{\epsilon^3} \cdot 1.728^n\right)$ for which the makespan is not greater than $(1+\epsilon)$ times the optimal makespan. The latter point denotes that this is an ϵ-approximation scheme. Our algorithm belongs to the class of moderate exponential-time approximation algorithms. Notice that the 3-machine flowshop problem does not admit an FPTAS (fully polynomial-time approximation scheme) because it is strongly NP-hard, meaning that no ϵ-approximation algorithm exists to solve the 3-machine flowshop in time $\mathcal{O}\left(poly(n, \frac{1}{\epsilon})\right)$ unless P = NP (Vazinari [33]). In comparison, Hall et al. [13] provide for the m-machine flowshop problem an FPT-AS (fixed-parameter tractable approximation scheme), namely an ϵ-approximation algorithm that runs in time $\mathcal{O}(f(\epsilon, \kappa) \cdot poly(n))$ for κ a fixed parameter of the instance and f a computable function. Hall et al. [13] choose κ to be the number of machines of the flowshop, leading to an FPT-AS that runs in time $\mathcal{O}\left(n^{3.5} \cdot (\frac{m}{\epsilon})^{\frac{m^4}{\epsilon^2}}\right)$. In our case, we should consider the case $m = 3$.

Input: $\epsilon > 0$, 3-machine flowshop on n jobs with processing times $\{p_{ij} : i \in [3], j \in [n]\}$
Output: solution at most $1+\epsilon$ times the optimal solution
$P = \max_{i \in [3], j \in [n]} \{p_{ij}\}$;
$K = \frac{\epsilon P}{n+2}$;
for $i \in [3], j \in [n]$ **do**
  $p'_{ij} = \lceil \frac{p_{ij}}{K} \rceil$;
end
Solve 3-machine flowshop on n jobs with new processing times $\{p'_{ij} : i \in [3], j \in [n]\}$ with Algorithm 5 that outputs permutation π';
Return π'

Algorithm 6: Hybrid approximation scheme for the 3-machine flowshop

Lemma 4.19. *Let π^* be an optimal solution of the 3-machine flowshop problem, for the processing*

times $\{p_{ij} : i \in [3], j \in [n]\}$. *Let* π' *be the output of Algorithm 6. We have*

$$C_{max}(\pi') \leq (1+\epsilon) \cdot C_{max}(\pi^*)\,.$$

Next, we introduce two observations necessary to prove Lemma 4.19. The proofs are omitted because of their simplicity.

Observation 4.20. *Let* π *be a permutation and let* α *be a non-negative real number. We note* $C_{max}(\pi)$ *the makespan of* π *of the 3-machine flowshop for processing times* $\{p_{ij} : i \in [3], j \in [n]\}$. *We note* $C'_{max}(\pi)$ *the makespan of* π *of the 3-machine flowshop for processing times* $\{p'_{ij} : i \in [3], j \in [n]\}$ *such that* $p'_{ij} := \alpha p_{ij}$ *for all* i, j. *Then,*

$$C'_{max}(\pi) = \alpha C_{max}(\pi)\,.$$

Notice that for $p'_{ij} \leq \alpha p_{ij}$, *we have* $C'_{max}(\pi) \leq \alpha C_{max}(\pi)$ *even if the critical path in* π *may differ to obtain* C_{max} *and* C'_{max}.

Observation 4.21. *Let* π *be a permutation and let* β *be a real number such that* $\beta \geq -\min_{i\in[3],j\in[n]}\{p_{ij}\}$. *We note* $C_{max}(\pi)$ *the makespan of* π *of the 3-machine flowshop for processing times* $\{p_{ij} : i \in [3], j \in [n]\}$. *We note* $C''_{max}(\pi)$ *the makespan of* π *of the 3-machine flowshop for processing times* $\{p''_{ij} : i \in [3], j \in [n]\}$ *such that* $p''_{ij} := p_{ij} + \beta$ *for all* $i \in [3], j \in [n]$. *Then,*

$$C''_{max}(\pi) \leq C_{max}(\pi) + \beta(n+2)\,.$$

Notice that for $p''_{ij} \leq p_{ij} + \beta$, *we still have* $C''_{max}(\pi) \leq C_{max}(\pi) + \beta(n+2)$ *even if the critical path in* π *may differ to obtain* C_{max} *and* C''_{max}.

Proof of Lemma 4.19. Given $\epsilon > 0$, let us prove Lemma 4.19. The new processing times considered $p'_{ij} := \lceil \frac{p_{ij}}{K} \rceil$ imply that $\frac{p_{ij}}{K} \leq p'_{ij} < \frac{p_{ij}}{K} + 1$. We note $C'_{\max}$ the makespan of the new problem, i.e. the 3-machine flowshop problem with processing times $\{p'_{ij} : i \in [3], j \in [n]\}$.

On the one hand, we have $p'_{ij} < \frac{p_{ij}}{K} + 1$ for all $i \in [3], j \in [n]$. Thus, according to Observations 4.20 and 4.21 considering the optimal permutation π^*,

$$C'_{\max}(\pi^*) \leq \frac{C_{\max}(\pi^*)}{K} + n + 2\,,$$

namely, because $K > 0$,

$$KC'_{\max}(\pi^*) \leq C_{\max}(\pi^*) + K(n+2)\,. \tag{14}$$

On the other hand, we have $\frac{p_{ij}}{K} \leq p'_{ij}$. Thus, according to Observation 4.20 considering the output permutation π' of Algorithm 6,

$$\frac{C_{\max}(\pi')}{K} \leq C'_{\max}(\pi')\,,$$

namely, because $K > 0$,

$$\begin{aligned}
C_{\max}(\pi') &\leq KC'_{\max}(\pi') && (15)\\
&\leq KC'_{\max}(\pi^*) && (16)\\
&\leq C_{\max}(\pi^*) + K(n+2) = C_{\max}(\pi^*) + \epsilon P && (17)\\
&\leq C_{\max}(\pi^*) + \epsilon C_{\max}(\pi^*) = (1+\epsilon)C_{\max}(\pi^*)\,, && (18)
\end{aligned}$$

where (16) comes from the fact that π' is the optimal solution for makespan $C'_{\max}$, (17) results from Equation (14), and (18) is true because the makespan is always larger than $P = \max\limits_{i\in[3],j\in[n]}\{p_{ij}\}$. □

Theorem 4.22. *Algorithm 6 is an approximation scheme for the 3-machine flowshop problem and outputs a solution whose makespan it at most* $(1+\epsilon)$ *times the optimal value in time*

$$\mathcal{O}^*\left(\frac{1}{\epsilon^3}\cdot 1.728^n\right)\,.$$

Proof. First, according to Lemma 4.19, Algorithm 6 outputs a solution whose makespan it at most $(1+\epsilon)$ times the optimal value.

Second, Algorithm 5 solves the new problem in time $\mathcal{O}^*((\sum p'_{ij})^4 \cdot 1.728^n) = \mathcal{O}^*(\frac{1}{\epsilon^4}\cdot 1.728^n)$. Indeed,

$$\begin{aligned}
\sum p'_{ij} \leq \sum(\frac{p_{ij}}{K}+1) &= \frac{1}{K}\sum p_{ij} + 3n\\
&\leq \frac{1}{K}\cdot 3nP + 3n\\
&= \frac{3n(n+2)}{\epsilon} + 3n\,.
\end{aligned}$$

Thus, $\sum p'_{ij} \leq poly(n, \frac{1}{\epsilon})$. □

5 Conclusion

In this work, we extend the quantum-classical algorithm of Ambainis et al. [3] to optimization problems that satisfy Dynamic Programming Across the Subsets (DPAS) properties. We illustrate our hybrid algorithm Q-DDPAS on several NP-hard single-machine scheduling problems and adapt it for the 3-machine flowshop problem. Q-DDPAS reduces the best-known classical time complexity, often equal to $\mathcal{O}^*(2^n)$ for single-machine problems and $\mathcal{O}^*(3^n)$ for the 3-machine flowshop, to $\mathcal{O}^*(1.728^n)$, sometimes at the cost of an additional pseudo-polynomial factor as summarized in Table 1. Future work should be dedicated to finding a quantum brick, e.g. Grover Search (Grover [12]), Quantum Fourier Transform (Kitaev [15]) or Quantum Walks (Aharonov et al. [1]), that could speedup exponential algorithms such as Sort-and-Search (Lenté et al. [18]), Inclusion-Exclusion (Ploton [24]) or Branch-and-Reduce (T'kindt et al. [32]).

Acknowledgements

This work has been partially financed by the ANRT (Association Nationale de la Recherche et de la Technologie) through the PhD number 2021/0281 with CIFRE funds.

References

[1] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani, *Quantum walks on graphs*, Proceedings of the thirty-third annual ACM symposium on Theory of computing, 2001, pp. 50–59.

[2] Jonathan Allcock, Yassine Hamoudi, Antoine Joux, Felix Klingelhöfer, and Miklos Santha, *Classical and quantum algorithms for variants of subset-sum via dynamic programming*, 30th Annual European Symposium on Algorithms (ESA 2022), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[3] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs, *Quantum speedups for exponential-time dynamic programming algorithms*,

Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2019, pp. 1783–1793.

[4] Charles H Bennett, *Logical reversibility of computation*, IBM journal of Research and Development **17** (1973), no. 6, 525–532.

[5] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al., *Variational quantum algorithms*, Nature Reviews Physics **3** (2021), no. 9, 625–644.

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

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

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, *A quantum approximate optimization algorithm*, arXiv preprint arXiv:1411.4028 (2014).

[9] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: a survey*, Annals of discrete mathematics, vol. 5, Elsevier, 1979, pp. 287–326.

[10] Camille Grange, Eric Bourreau, Michael Poss, and Vincent T'Kindt, *Quantum speed-ups for single-machine scheduling problems*, 2023 Genetic and Evolutionary Computation Conference (GECCO 2023), ACM, 2023.

[11] Camille Grange, Michael Poss, and Eric Bourreau, *An introduction to variational quantum algorithms for combinatorial optimization problems*, 4OR (2023), 1–41.

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

[13] Leslie A Hall, *Approximability of flow shop scheduling*, Mathematical Programming **82** (1998), no. 1-2, 175–190.

[14] Michael Held and Richard M Karp, *The traveling-salesman problem and minimum spanning trees*, Operations Research **18** (1970), no. 6, 1138–1162.

[15] A Yu Kitaev, *Quantum measurements and the abelian stabilizer problem*, arXiv preprint quant-ph/9511026 (1995).

[16] Krzysztof Kurowski, Tomasz Pecyna, Mateusz Slysz, Rafał Różycki, Grzegorz Waligóra, and Jan Weglarz, *Application of quantum approximate optimization algorithm to job shop scheduling problem*, European Journal of Operational Research (2023).

[17] 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** (1990), 125–133.

[18] Christophe Lenté, Mathieu Liedloff, Ameur Soukhal, and Vincent T'kindt, *On an extension of the sort & search method with application to scheduling theory*, Theoretical Computer Science **511** (2013), 13–22.

[19] Masayuki Miyamoto, Masakazu Iwamura, Koichi Kise, and François Le Gall, *Quantum speedup for the minimum steiner tree problem*, Computing and Combinatorics: 26th International Conference, COCOON 2020, Atlanta, GA, USA, August 29–31, 2020, Proceedings, Springer, 2020, pp. 234–245.

[20] Giacomo Nannicini, *Performance of hybrid quantum-classical variational heuristics for combinatorial optimization*, Physical Review E **99** (2019), no. 1, 013304.

[21] ———, *Fast quantum subroutines for the simplex method*, Operations Research (2022), In press.

[22] Michael A Nielsen and Isaac L Chuang, *Quantum computation and quantum information*, Cambridge university press, 2010.

[23] Michael L Pinedo, *Scheduling*, vol. 29, Springer, 2012.

[24] Olivier Ploton, *Contributions of inclusion-exclusion to exact or approximate solution of scheduling problems*, 2023.

[25] Olivier Ploton and Vincent T'kindt, *Exponential-time algorithms for parallel machine scheduling problems*, Journal of Combinatorial Optimization **44** (2022), no. 5, 3405–3418.

[26] ———, *Moderate worst-case complexity bounds for the permutation flowshop scheduling problem using inclusion–exclusion*, Journal of Scheduling **26** (2023), no. 2, 137–145.

[27] Yue Ruan, Samuel Marsh, Xilin Xue, Zhihao Liu, Jingbo Wang, et al., *The quantum approximate algorithm for solving traveling salesman problem*, Computers, Materials & Continua **63** (2020), no. 3, 1237–1247.

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

[29] Kazuya Shimizu and Ryuhei Mori, *Exponential-time quantum algorithms for graph coloring problems*, Algorithmica (2022), 1–19.

[30] David Sutter, Giacomo Nannicini, Tobias Sutter, and Stefan Woerner, *Quantum speedups for convex dynamic programming*, arXiv preprint arXiv:2011.11654 (2020).

[31] Zsolt Tabi, Kareem H El-Safty, Zsófia Kallus, Péter Hága, Tamás Kozsik, Adam Glos, and Zoltán Zimborás, *Quantum optimization for the graph coloring problem with space-efficient embedding*, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2020, pp. 56–62.

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

[33] Vijay V Vazirani, *Approximation algorithms*, vol. 1, Springer, 2001.

A Notations and upper bounds

In what follows, we use the notation

$$\binom{n}{\leq k} = \sum_{i=1}^{k} \binom{n}{i}.$$

We also define the binary entropy of $\epsilon \in]0,1[$ by $H(\epsilon) = -(\epsilon \log_2(\epsilon) + (1-\epsilon)\log_2(1-\epsilon))$. We remind some useful upper bounds of binomial coefficients [3]:

$$\forall k \in [\![1, n]\!], \quad \binom{n}{k} \leq 2^{H\left(\frac{k}{n}\right)\cdot n},$$

$$\forall k \in \left[\!\left[1, \frac{n}{2}\right]\!\right], \quad \binom{n}{\leq k} \leq 2^{H\left(\frac{k}{n}\right)\cdot n}.$$

Observe that $\binom{n}{\leq n/4}$ is bounded above by $2^{H\left(\frac{n/4}{n}\right)\cdot n}$, where

$$H\left(\frac{n/4}{n}\right) = H\left(\frac{1}{4}\right) = -\left(\frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{3}{4}\log_2\left(\frac{3}{4}\right)\right) = 0.811\,,$$

leading to

$$\binom{n}{\leq n/4} \leq 2^{0.811n}. \tag{19}$$

In the same way, we can show that

$$\binom{n}{\leq 0.945 \cdot n/4} \leq 2^{0.789n}. \tag{20}$$

Similarly, $\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}}$ is bounded above by

$$\sqrt{2^{H\left(\frac{n/4}{n/2}\right)\cdot\frac{n}{2}} 2^{H\left(\frac{n/2}{n}\right)\cdot n}} = 2^{\frac{1}{2}\left(\frac{1}{2}H\left(\frac{1}{2}\right)+H\left(\frac{1}{2}\right)\right)\cdot n} = 2^{\frac{3}{4}H\left(\frac{1}{2}\right)n},$$

where $H\left(\frac{1}{2}\right) = 1$, leading to

$$\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}} \leq 2^{0.75n}. \tag{21}$$

In the same way, we can show that

$$\sqrt{\binom{n}{n/2}\binom{n/2}{n/4}\binom{n/4}{0.945\cdot n/4}} \leq 2^{0.789n}. \tag{22}$$

B Omitted proof

In this appendix, we detail the proof of Lemma 3.25. Next, we compute $U_{\text{recur}}U_{\text{ini}}|\text{ini}\rangle$ and show that $\text{OPT}[[n], 0]$ is stored in register of indexes I_2^6 with high probability. We write the following computations as if the algorithm QMF was returning the optimal solution with probability 1. First,

we compute $U_{\text{ini}}\left|\text{ini}\right\rangle$.

$$
\begin{aligned}
U_{\Lambda_{\text{add}}}^{I^1\oplus I^2\oplus I^4}\left|\text{ini}\right\rangle =& U_{\Lambda_{\text{add}}}^{I^1\oplus I^2\oplus I^4}\underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\underbrace{\left|0\right\rangle^{\otimes 3}}_{I^2}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^3}\underbrace{\left|0\right\rangle^{\otimes 3}}_{I^4}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^5}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^6}\\
=& \underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\\
&\underbrace{\left|\lambda_1^s\right\rangle\left|\lambda_1^t\right\rangle\left|0\right\rangle}_{I^2}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^3}\underbrace{\left|\lambda_2^s\right\rangle\left|\lambda_2^t\right\rangle\left|0\right\rangle}_{I^4}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^5}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^6}.
\end{aligned}
$$

Thus,

$$
\begin{aligned}
U_{\text{ini}}\left|\text{ini}\right\rangle =& (U_{\Omega_{\text{add}}}^{I^2}\otimes U_{\Omega_{\text{add}}}^{I^4})\cdot U_{\Lambda_{\text{add}}}^{I^1\oplus I^2\oplus I^4}\left|\text{ini}\right\rangle\\
=& (U_{\Omega_{\text{add}}}^{I^2}\otimes U_{\Omega_{\text{add}}}^{I^4})\underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\underbrace{\left|\lambda_1^s\right\rangle\left|\lambda_1^t\right\rangle\left|0\right\rangle}_{I^2}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^3}\underbrace{\left|\lambda_2^s\right\rangle\left|\lambda_2^t\right\rangle\left|0\right\rangle}_{I^4}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^5}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^6}\\
=& \underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\underbrace{\left|\lambda_1^s\right\rangle}_{I_1^2}\underbrace{\left|\lambda_1^t\right\rangle}_{I_2^2}\left(\sum_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{\left|\omega\right\rangle}_{I_3^2}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^3}\right)\\
&\underbrace{\left|\lambda_2^s\right\rangle}_{I_1^4}\underbrace{\left|\lambda_2^t\right\rangle}_{I_2^4}\left(\sum_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{\left|\omega\right\rangle}_{I_3^4}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^5}\right)\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^6}.
\end{aligned}
$$

Second, we apply the tensor product of the two first QMF to the previous state.

$$
\begin{aligned}
&\left(U_{\text{QMF}}^{I_3^2\oplus I^3}[U_a^{I_3^2}]\otimes U_{\text{QMF}}^{I_3^4\oplus I^5}[U_a^{I_3^4}]\right)\underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\\
&\underbrace{\left|\lambda_1^s\right\rangle}_{I_1^2}\underbrace{\left|\lambda_1^t\right\rangle}_{I_2^2}\left(\sum_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{\left|\omega\right\rangle}_{I_3^2}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^3}\right)\\
&\underbrace{\left|\lambda_2^s\right\rangle}_{I_1^4}\underbrace{\left|\lambda_2^t\right\rangle}_{I_2^4}\left(\sum_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{\left|\omega\right\rangle}_{I_3^4}\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^5}\right)\underbrace{\left|0\right\rangle^{\otimes 2}}_{I^6}\\
=& \underbrace{\left|[n]\right\rangle\left|0\right\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\\
&\underbrace{\left|\lambda_1^s\right\rangle}_{I_1^2}\underbrace{\left|\lambda_1^t\right\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{\left|\omega\right\rangle}_{I_3^2}\underbrace{\left|\arg\min_{\omega} r(\omega)\right\rangle\left|\min_{\omega} r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}
\end{aligned}
$$

$$\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6}.$$

Thus, we apply the second circuit of QMF.

$$\begin{aligned}
U_{\text{recur}}U_{\text{ini}}|\text{ini}\rangle =&U_{\text{QMF}}^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1\oplus I^6}[U_{\text{recur1}}]U_{\text{ini}}|\text{ini}\rangle\\
=&U_{\text{QMF}}^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1 I^6}[U_a^{I_1^2\oplus I_2^3\oplus I_1^4\oplus I_2^5\oplus I_2^1}]\underbrace{|[n]\rangle|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}\\
&\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}\underbrace{|0\rangle^{\otimes 2}}_{I^6}\\
=&\underbrace{|[n]\rangle|0\rangle}_{I^1}\sum_{(\lambda_1^s,\lambda_1^t,\lambda_2^s,\lambda_2^t)\in\Lambda_{\text{add}}([n],0)}\frac{1}{\sqrt{|\Lambda_{\text{add}}([n],0)|}}\\
&\underbrace{|\lambda_1^s\rangle}_{I_1^2}\underbrace{|\lambda_1^t\rangle}_{I_2^2}\sum_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)|}}\underbrace{|\omega\rangle}_{I_3^2}\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^3\otimes I_2^3}\\
&\underbrace{|\lambda_2^s\rangle}_{I_1^4}\underbrace{|\lambda_2^t\rangle}_{I_2^4}\sum_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}\frac{1}{\sqrt{|\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)|}}\underbrace{|\omega\rangle}_{I_3^4}\\
&\underbrace{\left|\arg\min_\omega r(\omega)\right\rangle\left|\min_\omega r(\omega)\right\rangle}_{I_1^5\otimes I_2^5}\underbrace{\left|\arg\min_{\lambda\in\Lambda_{\text{add}}([n],0)} r(\lambda_1^s,\min_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}r(\omega),\lambda_2^s,\min_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}r(\omega),0)\right\rangle}_{I_1^6}\\
&\underbrace{\left|\min_{\lambda\in\Lambda_{\text{add}}([n],0)} r(\lambda_1^s,\min_{\omega\in\Omega_{\text{add}}(\lambda_1^s,\lambda_1^t)}r(\omega),\lambda_2^s,\min_{\omega\in\Omega_{\text{add}}(\lambda_2^s,\lambda_2^t)}r(\omega),0)\right\rangle}_{I_2^6}.
\end{aligned}$$

According to definition of a and recurrence (Add-D-DPAS), the results stored in register of indexes I_2^6 is $\text{OPT}[[n],0]$.

Notice that optimal permutation $\pi^*[[n],0]$ can be *rebuilt* with registers of indexes I_1^3, I_1^5 and I_1^6, and with the access to the results of the classical part in the QRAM.