On the Complexity of Mapping Pipelined Filtering Services on Heterogeneous Platforms

Anne Benoit, Fanny Dufossé and Yves Robert

LIP, École Normale Supérieure de Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France
{Anne.Benoit|Fanny.Dufosse|Yves.Robert}@ens-lyon.fr

Abstract

In this paper, we explore the problem of mapping filtering services on large-scale heterogeneous platforms. Two important optimization criteria should be considered in such a framework. The period, which is the inverse of the throughput, measures the rate at which data sets can enter the system. The latency measures the response time of the system in order to process one single data set entirely. Both criteria are antagonistic. For homogeneous platforms, the complexity of period minimization is already known [12]; we derive an algorithm to solve the latency minimization problem in the general case with service precedence constraints; we also show that the bi-criteria problem (latency minimization without exceeding a prescribed value for the period) is of polynomial complexity. However, when adding heterogeneity to the platform, we prove that minimizing the period or the latency becomes NP-complete, and that these problems cannot be approximated by any constant factor (unless P=NP). The latter results hold true even for services without precedence constraints.

Key words: *query optimization, web service, filter, workflow, period, latency, complexity results.*

1 Introduction

This paper deals with the problem of query optimization over web services [12, 6]. The problem is close to the problem of mapping pipelined workflows onto distributed architectures, but involves several additional difficulties due to the filtering properties of the services.

In a nutshell, pipelined workflows are a popular programming paradigm for streaming applications like video and audio encoding and decoding, DSP applications, etc [8, 13, 17]. A workflow graph contains several *nodes*, and these nodes are connected to each other using first-in-first-out *channels*. Data is input into the graph using input channel(s) and the outputs are produced on the output channel(s). The goal is to map each node onto some processor so as to optimize some scheduling objective. Since data continually flows through these applications, typical objectives of the scheduler are *throughput* maximization (or equivalently *period* minimization, where the period is defined as the inverse of the throughput) and/or *latency* (also called response time) minimization [14, 15, 5, 16]. Consider for instance a video and audio decoding application. We want that the video begin as fast as possible, i.e., we want to minimize the latency. Moreover, we need to respect the streaming bandwidth, i.e., there is a bound on the period that should not be violated. In such a case, the goal is thus to minimize the latency with a bound on the period.

In the query optimization problem, we have a collection of various services that must be applied on a stream of consecutive data sets. As for workflows, we have a graph with nodes (the services) and precedence edges (dependence constraints between services), with data flowing continuously from the input node(s) to the output node(s). Also, the goal is to map each service onto a processor, or server, so as to optimize the same objectives as before (period and/or latency). But in addition, services can *filter* the data by a certain amount, according to their *selectivity*. Consider a service C_i with selectivity σ_i: if the incoming data is of size δ, then the outgoing data will be of size $\delta \times \sigma_i$. The initial data is of size δ_0. We see that the data is shrunk by C_i (hence the term "filter") when $\sigma_i < 1$ but it can also be expanded if $\sigma_i > 1$. Each service has an elementary cost c_i, which represents the volume of computations required to process a data set of size δ_0. But the volume of computa-

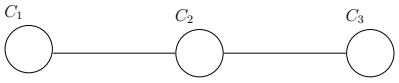

Figure 1. Chaining services.

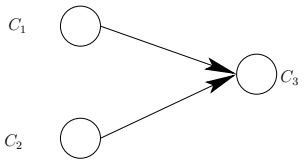

Figure 2. Combining selectivities

tions is proportional to the actual size of the input data, which may have shrunk or expanded by the predecessors of C_i in the mapping. Altogether, the time to execute a data set of size $\sigma \times \delta_0$ when service C_i is mapped onto server S_u of speed s_u is $\sigma \frac{c_i}{s_u}$. Here σ denotes the combined selectivity of all predecessor of C_i in the mapping.

Consider now two arbitrary services C_i and C_j. If there is a precedence constraint from C_i to C_j, we need to enforce it. But if there is none, meaning that C_i and C_j have no precedence constraints, we may still introduce a (fake) edge, say from C_j to C_i, in the mapping, meaning that the output of C_j is fed as input to C_i. If the selectivity of C_j is small ($\sigma_j < 1$), then it shrinks each data set, and C_i will operate on data sets of reduced volume. As a result, the cost of C_i will decrease in proportion to the volume reduction, leading to a better solution than running both services in parallel. Basically, there are two ways to decrease the final cost of a service: (i) map it on a fast server; and (ii) map it as a successor of a service with small selectivity. In general, we have to organize the execution of the application by assigning a server to each service and by deciding which service will be a predecessor of which other service (therefore building an execution graph, or *plan*), with the goal of minimizing the objective function. The edges of the execution graph must include all the original dependence edges of the application. We are free to add more edges if it decreases the objective function. Note that the selectivity of a service influences the execution time of *all* its successors, if any, in the mapping. For example if three services C_1, C_2 and C_3 are arranged along a linear chain, as in Figure 1, then the cost of C_2 is $\sigma_1 c_2$ and the cost of C_3 is $\sigma_1\sigma_2 c_3$. If C_i is mapped onto S_i, for $i = 1, 2, 3$, then the period is $\mathcal{P} = \max\left(\frac{c_1}{s_1}, \frac{\sigma_1 c_2}{s_2}, \frac{\sigma_1\sigma_2 c_3}{s_3}\right)$, while the latency is $\mathcal{L} = \frac{c_1}{s_1} + \frac{\sigma_1 c_2}{s_2} + \frac{\sigma_1\sigma_2 c_3}{s_3}$. Here, we also note that selectivities are independent: for instance if C_1 and C_2 are both predecessors of C_3, as in Figure 1 or in Figure 2, then the cost of C_3 becomes $\sigma_1\sigma_2 c_3$. With the mapping of Figure 2, the period is $\mathcal{P} = \max\left(\frac{c_1}{s_1}, \frac{c_2}{s_2}, \frac{\sigma_1\sigma_2 c_3}{s_3}\right)$, while the latency is $\mathcal{L} = \max\left(\frac{c_1}{s_1}, \frac{c_2}{s_2}\right) + \frac{\sigma_1\sigma_2 c_3}{s_3}$. We see from the latter formulas that the model neglects the cost of *joins* when combining two services as predecessors of a third one.

To give some concrete examples, we note that this model can be used when repeatedly searching regular expressions in a file. A service corresponds to a regular expression, and its selectivity is the probability to find this expression in a given record of the file. Another example is the sieve of Eratosthenes. Suppose that we know all prime numbers smaller than 100. To compute all prime numbers between 100 and 10000, we can associate a filtering service of selectivity $\frac{p-1}{p}$ to any prime number $p \leq 100$; such a service transmits a number to its successors if and only if this number is not a multiple of p. Then a number between 100 and 10000 incoming to a plan for this problem instance is transmitted through all services if and only if it is prime. Clearly, selectivities are independent and combine multiplicatively in this example.

All hypotheses and mapping rules of this paper are those of Srivastava et al. [12, 6]. Although their papers mainly deal with query optimization over web services (already an increasingly important application with the advent of Web Service Management Systems [9, 11]), the approach applies to general data streams [3] and to database predicate processing [7, 10]. Finally (and quite surprisingly), we note that our framework is quite similar to the problem of scheduling unreliable jobs on parallel machines [1] where service selectivities correspond to job failure probabilities.

As already pointed out, period and latency are both very important objectives. The inverse of the period (the throughput) measures the aggregate rate of processing of data, and it is the rate at which data sets can enter the system. The latency is the time elapsed between the beginning and the end of the execution of a given data set, hence it measures the response time of the system to process the data set entirely. Minimizing the latency is antagonistic to minimizing the period, and tradeoffs should be found between these criteria. Efficient mappings aim at the minimization of a single criterion, either the period or the latency, but they can also use a bicriteria approach, such as minimizing the latency under period constraints (or the converse). The main objective of this work is to assess the complexity of the previous

optimization problems, first with identical servers, and then with different-speed servers.

In this paper, we establish several new and important complexity results. First we introduce an optimal polynomial algorithm for the latency minimization problem on a homogeneous platform. This result nicely complements the corresponding result for period minimization, that was shown to have polynomial complexity in [12]. We also show the polynomial complexity of the bi-criteria problem (minimizing latency while not exceeding a threshold period). Moving to heterogeneous resources, we prove the NP-completeness of both the period and latency minimization problems, even for services without precedence constraints. Therefore, the bi-criteria problem also is NP-complete in this case. Furthermore, we prove that there exists no constant factor approximation algorithms for these problems unless P=NP. We also assess the complexity of several particular problem instances.

The rest of this paper is organized as follows. First we formally state the optimization problems that we address in Section 2. Next we detail two little examples aimed at showing the intrinsic combinatorial complexity of the problem (Section 3). Then Section 4 is devoted to problems with identical resources (homogeneous platforms), while Section 5 is the counterpart for different-speed processors (heterogeneous platforms). Finally we give some conclusions and perspectives in Section 6.

2 Framework

As stated above, the target application $\mathcal{A}$ is a set of services (or filters, or queries) linked by precedence constraints. We write $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ where $\mathcal{F} = \{C_1, C_2, \ldots, C_n\}$ is the set of services and $\mathcal{G} \subset \mathcal{F} \times \mathcal{F}$ is the set of precedence constraints. If $\mathcal{G} = \emptyset$, we have services without precedence constraints. A service C_i is characterized by its cost c_i and its selectivity σ_i.

For the computing resources, we have a set $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ of servers. In the case of homogeneous platforms, servers are identical while in the case of heterogeneous platforms, each server S_u is characterized by its speed s_u. We always assume that there are more servers available than services (hence $n \leq p$), and we search a one-to-one mapping, or allocation, of services to servers. The one-to-one allocation function alloc associates to each service C_i a server $S_{\mathsf{alloc}(i)}$.

We also have to build a graph $G = (\mathcal{C}, \mathcal{E})$ that summarizes all precedence relations in the mapping. The nodes of the graph are couples $(C_i, S_{\mathsf{alloc}(i)}) \in \mathcal{C}$, and thus define the allocation function. There is an arc $(C_i, C_j) \in \mathcal{E}$ if C_i precedes C_j in the execution. There are two types of such arcs: those induced by the set of precedence constraints $\mathcal{G}$, which must be enforced in any case, and those added to reduce the period or the latency. $\mathsf{Ancest}_j(G)$ denotes the set of all ancestors[1] of C_j in G, but only arcs from direct predecessors are kept in $\mathcal{E}$. In other words, if $(C_i, C_j) \in \mathcal{G}$, then we must have $C_i \in \mathsf{Ancest}_j(G)$ [2]. The graph G is called a plan. Given a plan G, the execution time of a service C_i is $cost_i(G) = \left(\prod_{C_j \in \mathsf{Ancest}_i(G)} \sigma_j\right) \times \frac{c_i}{s_{\mathsf{alloc}(i)}}$. We note $L_G(C_i)$ the completion time of service C_i with the plan G, which is the length of the path from an entry node to C_i, where each node is weighted with its execution time. We can now formally define the period $\mathcal{P}$ and the latency $\mathcal{L}$ of a plan G:

$$\mathcal{P}(G) = \max_{(C_i, S_u) \in \mathcal{C}} cost_i(G)$$

and

$$\mathcal{L}(G) = \max_{(C_i, S_u) \in \mathcal{C}} L_G(C_i).$$

In the following we study three optimization problems: (i) MINPERIOD: find a plan G that minimizes the period; (ii) MINLATENCY: find a plan G that minimizes the latency; and (iii) BICRITERIA: given a bound on the period K, find a plan G whose period does not exceed K and whose latency is minimal. Each of these problems can be tackled, (a) either with an arbitrary precedence graph $\mathcal{G}$ (case PREC) or without (case NOPREC); and (b) either with identical servers ($s_u = s$ for all servers S_u, homogeneous case HOM), or with different-speed servers (heterogeneous case HET). For instance, MINPERIOD-NOPREC-HOM is the problem of minimizing the period for services without precedence constraints on homogeneous platforms while MINLATENCY-PREC-HET is the problem of minimizing the latency for arbitrary precedence constraints on heterogeneous platforms.

3 Working out examples

In this section we deal with two little examples. The first one considers services without precedence constraints and different-speed processors (hence a problem NOPREC-HET), while the second one involves precedence constraints and identical resources (PREC-HOM).

[1] The ancestors of a service are the services preceding it, and the predecessors of their predecessors, and so on.

[2] Equivalently, $\mathcal{G}$ must be included, in the transitive closure of $\mathcal{E}$.

3.1 An example for the NoPrec-Het problem

Consider a problem instance with three services C_1, C_2 and C_3 without precedence constraints. Assume that $c_1 = 1$, $c_2 = 4$, $c_3 = 10$, and that $\sigma_1 = \frac{1}{2}$, $\sigma_2 = \sigma_3 = \frac{1}{3}$. Suppose that we have three servers of respective speeds $s_1 = 1$, $s_2 = 2$ and $s_3 = 3$. What is the mapping which minimizes the period? and same question for the latency? We have to decide for an assignment of services to servers, and to build the best plan.

For MinPeriod-NoPrec-Het (period optimization), we can look for a plan with a period smaller than or equal to 1. In order to obtain an execution time smaller than or equal to 1 for service C_3, we need the selectivity of C_1 and C_2, and either server S_2 or server S_3. Server S_2 is fast enough to render the time of C_3 smaller than 1, so we decide to assign C_3 to S_2. Service C_2 also needs the selectivity of C_1 and a server of speed strictly greater than 1 to obtain an execution time less than 1. Thus, we assign C_1 to S_1 and make it a predecessor of C_2. In turn we assign C_2 to S_3 and make it a predecessor of C_3. We obtain a period of $\min\left(\frac{1}{1}, \frac{1}{2}\frac{4}{3}, \frac{1}{2\times 3}\frac{10}{2}\right) = 1$. It is the optimal solution. In this plan, the latency is equal to $1 + \frac{4}{6} + \frac{10}{12} = \frac{5}{2}$.

For MinLatency-NoPrec-Het (latency optimization), we have a first bound: $\frac{5}{2}$. Because of its cost, service C_3 needs at least one predecessor. If C_1 is the only predecessor of C_3, we have to assign C_3 to S_3 in order to keep the latency under $\frac{5}{2}$. The fastest computation time that we can then obtain for C_3 is $\frac{1}{2} + \frac{1}{2}\frac{10}{3}$, with C_1 assigned to S_2. In this case, the fastest completion time for C_2 is $\frac{5}{2}$: this is achieved by letting C_2 be a successor of C_1 in parallel with C_3. Suppose now that C_2 is a predecessor of C_3, and that there is an optimal solution in which C_2 is the only predecessor of C_3. Independently of the choice of the servers assigned to C_1 and C_2, if we put C_1 without any predecessor, it will end before C_2. So, we can make it a predecessor of C_3 without increasing its completion time. So, we are looking for a solution in which C_1 and C_2 are predecessors of C_3. There are three possibilities left: (i) C_1 is a predecessor of C_2; (ii) C_2 is a predecessor of C_1; and (iii) C_1 and C_2 have no predecessors. In the first two cases, we compute for each service a cost weighted by the product of the selectivities of its predecessors. Then, we associate the fastest server to the service with the longest weighted cost and so on. We obtain $\frac{5}{2}$ in both cases. For the last case, we know that the real cost of C_1 will have no influence on the latency, hence we assign it to the slowest server S_1. The weighted cost of the remaining services is 4 for C_2 and $\frac{10}{6}$ for C_3. So, we assign S_3 to C_2 and S_2 to C_3. We obtain a latency of $\frac{4}{3} + \frac{1}{2\times 3}\frac{10}{2} = \frac{13}{6}$. We cannot obtain a strictly faster solution if C_2 is not a predecessor of C_3. As a result, $\frac{13}{6}$ is the optimal latency. In this optimal plan for the latency, the period is $\frac{4}{3}$.

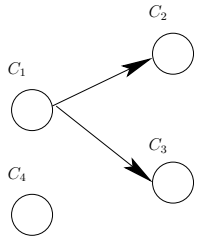

Figure 3. Precedence constraints.

3.2 An example for the Prec-Hom problem

Let $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ be the following set of 4 services: $c_1 = c_2 = 1$, $c_3 = c_4 = 4$, $\sigma_1 = \frac{1}{2}$, $\sigma_2 = \frac{4}{5}$, $\sigma_3 = \sigma_4 = 2$ and $\mathcal{G} = \{(C_1, C_2), (C_1, C_3)\}$ (see Figure 3). With this dependence set, we have 3 possible combinations for ordering C_1, C_2, C_3, and for each of these orderings, 10 possible graphs when adding C_4. We target a homogeneous platform with four identical servers of speed $s = 1$.

For MinPeriod-Prec-Hom, suppose that we can obtain a period strictly less than 2. C_1 is the only service that can be placed without predecessor, because $c_4 > 2$, and both C_2 and C_3 need C_1 as an ancestor (precedence constraints). C_2 is the only remaining service of cost strictly less than 4. It can be placed with C_1 as unique predecessor. Then we place C_3 and C_4 with predecessors C_1 and C_2. We obtain a period $P = \frac{8}{5}$ (see Figure 4), which is optimal.

Let us study MinLatency-Prec-Hom. With the plan shown in Figure 5, we obtain a latency $L = 1 + \frac{1}{2} \times 4 = 3$. Suppose that we can obtain a latency strictly less than 3. Again, C_1 is the only service that can be placed without any predecessor. As for MinPeriod, C_2 is the

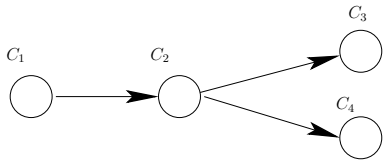

Figure 4. Optimal plan for period.

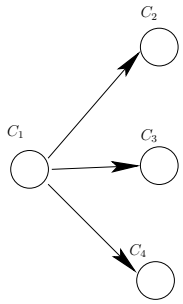

Figure 5. Optimal plan for latency.

only service that can be placed after C_1. But in this case, C_3 and C_4 cannot be placed after C_2, because it would give a latency $L = 1 + \frac{1}{2} \times 1 + \frac{1}{2} \times \frac{4}{5} \times 4 > 3$. Therefore, 3 is the optimal latency for this problem instance.

4 Homogeneous platforms

In this section we investigate the optimization problems with homogeneous resources. Problem MINPERIOD-PREC-HOM (minimizing the period with precedence constraints and identical resources) was shown to have polynomial complexity in [12, 6]. We show that Problem MINLATENCY-PREC-HOM is polynomial too. Because the algorithm is quite complicated, we start with an optimal algorithm for the simpler problem MINLATENCY-NOPREC-HOM. Although the polynomial complexity of the latter problem is a consequence of the former, it is insightful to follow the derivation for services without precedence constraints before dealing with the general case. Finally, we show that both BICRITERIA-NOPREC-HOM and BICRITERIA-PREC-HOM can also be solved in polynomial time.

4.1 Latency

We describe here optimal algorithms for MINLATENCY-HOM, without dependences first, and then for the general case.

Theorem 1. (Services without precedence constraints) *Algorithm 1 computes the optimal plan for* MINLATENCY-NOPREC-HOM *in time* $O(n^2)$.

Data: n services with selectivities $\sigma_1, ..., \sigma_p \leq 1$ without precedence constraints, $\sigma_{p+1}, ..., \sigma_n > 1$, and ordered costs $c_1 \leq \cdots \leq c_p$
Result: a plan G optimizing the latency

```
G is the graph reduced to node C_1;
for i = 2 to n do
    for j = 0 to i − 1 do
        Compute the completion time L_j(C_i) of C_i in
        G with predecessors C_1, ..., C_j;
    end
    Choose j such that L_j(C_i) = min_k{L_k(C_i)};
    Add the node C_i and the edges C_1 → C_i, ...,
    C_j → C_i to G;
end
```

Algorithm 1: Optimal algorithm for MINLATENCY-NOPREC-HOM.

Proof. We show that Algorithm 1 verifies the following properties:

- (A) $L_G(C_1) \leq L_G(C_2) \leq \cdots \leq L_G(C_p)$
- (B) $\forall i \leq n$, $L_G(C_i)$ is optimal

Because the latency of any plan G' is the completion time of its last node (a node C_i such that $\forall C_j, L_{G'}(C_i) \geq L_{G'}(C_j)$), property (B) shows that $\mathcal{L}(G)$ is the optimal latency. We prove properties (A) and (B) by induction on i: for every i we prove that $L_G(C_i)$ is optimal and that $L_G(C_1) \leq L_G(C_2) \leq \cdots \leq L_G(C_i)$.

For $i = 1$: C_1 has no predecessor in G, so $L_G(C_1) = c_1$. Suppose that there exists G' such that $L_{G'}(C_1) < L_G(C_1)$. If C_1 has no predecessor in G', then $L_{G'}(C_1) = c_1 = L_G(C_1)$. Otherwise, let C_i be a predecessor of C_1 in G' such that C_i has no predecessor itself. $L_{G'}(C_1) > c_i \geq c_1$. In both cases, we obtain a contradiction with the hypothesis $L_{G'}(C_1) < L_G(C_1)$. So $L_G(C_1)$ is optimal.

Suppose that for a fixed $i \leq p$, $L_G(C_1) \leq L_G(C_2) \leq \cdots \leq L_G(C_{i-1})$ and $\forall j < i$, $L_G(C_j)$ is optimal. Suppose that there exists G' such that $L_{G'}(C_i)$ is optimal. Let C_k be the predecessor of C_i of greatest cost in G'. If $c_k > c_i$, we can choose in G' the same predecessors for C_i than for C_k, thus strictly reducing $L_{G'}(C_i)$. However, $L_{G'}(C_i)$ is optimal. So, we obtain a contradiction and $c_k \leq c_i$. Thus,

$$\begin{aligned} L_{G'}(C_i) &= L_{G'}(C_k) + \left(\prod_{C_j \in \mathsf{Ancest}_{L_{G'}(C_i)}} \sigma_j\right) c_i \\ &\geq L_{G'}(C_k) + \left(\prod_{j \leq k} \sigma_j\right) c_i \\ &\quad (\text{ by definition of } C_k) \\ &\geq L_G(C_i) \\ &\quad (\text{by construction of } G) \end{aligned}$$

Therefore, since $L_{G'}(C_i)$ is optimal by hypothesis, we have $L_{G'}(C_i) = L_G(C_i)$.
Suppose now that $L_G(C_i) < L_G(C_{i-1})$. Then, C_{i-1} is not a predecessor of C_i in G. We construct G'' such that all edges are the same as in G except those oriented to C_{i-1}: predecessors of C_{i-1} will be the same as predecessors of C_i. We obtain

$$\begin{array}{rcl} L_{G''}(C_{i-1}) & = & \max_{k\leq j} L_G(C_k) + \prod_{k\leq j} \sigma_k c_{i-1} \\ & & (\text{ by construction of node } C_{i-1}) \\ & \leq & \max_{k\leq j} L_G(C_k) + \prod_{k\leq j} \sigma_k c_i \\ & = & L_G(C_i) \end{array}$$

However, $L_G(C_{i-1})$ is optimal, and so $L_G(C_{i-1}) \leq L_{G''}(C_{i-1}) \leq L_G(C_i)$, which leads to a contradiction. Therefore, $L_G(C_1) \leq L_G(C_2) \leq \cdots \leq L_G(C_i)$.

At this point, we have proved that the placement of all services of selectivity smaller than 1 is optimal, and that $L_G(C_1) \leq L_G(C_2) \leq \cdots \leq L_G(C_p)$. We now proceed with services C_{p+1} to C_n.

Suppose that for a fixed $i > p$, $\forall j < i$, $L_G(C_j)$ is optimal. For all $k > p$, we have

$$\begin{array}{rcl} & & \max_{j\leq k} L_G(C_j) + \prod_{j\leq k} \sigma_j * c_i \\ & = & \max_{j=p}^{k} L_G(C_j) + \prod_{j=1}^{k} \sigma_j * c_i \\ & \geq & L_G(C_p) + \prod_{j\leq k} \sigma_j * c_i \\ & > & L_G(C_p) + \prod_{j\leq p} \sigma_j * c_i \end{array}$$

This relation proves that in G, service C_i has no predecessor of selectivity strictly greater than 1. Suppose that there exists G' such that $L_{G'}(C_i)$ is optimal. Let C_k be the predecessor of C_i in G' of greatest cost. Then $\mathsf{Ancest}_i(G') \in \{1, k\}$ and, similarly for the case $i \leq p$, we obtain $L_{G'}(C_i) \geq L_G(C_i)$, and thus $L_G(C_i)$ is optimal. □

Theorem 2. (General case) *Algorithm 2 computes the optimal plan for* MINLATENCY-PREC-HOM *in time* $O(n^6)$.

Data: n services, a set $\mathcal{G}$ of dependence constraints
Result: a plan G optimizing the latency
1 G is the graph reduced to the node C of minimal cost with no predecessor in $\mathcal{G}$;
2 **for** $i = 2$ **to** n **do**
3 // At each step we add one service to G, hence the $n-1$ steps;
4 Let S be the set of services not yet in G and such that their set of predecessors in $\mathcal{G}$ is included in G;
5 **for** $C \in S$ **do**
6 **for** $C' \in G$ **do**
7 Compute the set S' minimizing the product of selectivities among services of latency less than $L_G(C')$, and including all predecessors of C in $\mathcal{G}$ (using an algorithm from [6], whose execution time is $O(n^3)$);
8 **end**
9 Let S_C be the set that minimizes the latency of C in G and L_C be this latency;
10 **end**
11 Choose a service C such that $L_C = \min\{L_{C'}, C' \in S\}$;
12 Add to G the node C, and $\forall C' \in S_C$, the edge $C' \rightarrow C$;
13 **end**

Algorithm 2: Optimal algorithm for MINLATENCY-PREC-HOM.

Proof. Let $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ with $\mathcal{F} = \{C_1, C_2, \ldots, C_n)$ be an instance of MINLATENCY-PREC-HOM. Let G be the plan produced by Algorithm 2 on this instance, and services are renumbered so that C_i is the service added at step i of the algorithm. Then we prove by induction on i that $L_G(C_1) \leq L_G(C_2) \leq ... \leq L_G(C_n)$, and G is optimal for $L(C_i)$, $1 \leq i \leq n$. In the following, we say that a plan is *valid* if all precedence edges are included. The plan G is valid by construction of the algorithm.

By construction, C_1 has no predecessors in G. Therefore, $L_G(C_1) = c_1$. Let G' be a valid plan such that $L_{G'}(C_1)$ is optimal. If C_1 has no predecessors in G', then $L_{G'}(C_1) = L_G(C_1)$. Otherwise, let C_i be a predecessor of C_1 which has no predecessors in G'. G' is valid, thus C_i has no predecessors in $\mathcal{G}$. And by construction of G, we have $c_1 \leq c_i$. Therefore, $L_{G'}(C_1) \geq c_i \geq c_1 = L_G(C_1)$. Since $L_{G'}(C_1)$ is optimal, $L_G(C_1) = L_{G'}(C_1)$ and thus $L_G(C_1)$ is optimal.

Suppose that for a fixed $i \leq n$, we have $L_G(C_1) \leq L_G(C_2) \leq ... \leq L_G(C_{i-1})$, and $\forall j < i, L_G(C_j)$ is optimal. Let us prove first that $L_G(C_{i-1}) \leq L_G(C_i)$. If C_{i-1} is a predecessor of C_i, then the result is true. Otherwise, and if $L_G(C_{i-1}) > L_G(C_i)$, then C_i would have been chosen at step $i-1$ of the algorithm (line 9)

instead of C_{i-1}, which leads to a contradiction. It remains to prove that $L_G(C_i)$ is optimal. Let us consider a valid plan G' such that $L_{G'}(C_i)$ is optimal.
(i) Suppose first that C_i has at least one predecessor C_l with $l > i$ in G'. For such predecessors, at least one of them has its own set of predecessors included in $\{C_1, ..., C_{i-1}\}$. Let C_k be the service of maximal latency $L_{G'}(C_k)$ of the previous set of predecessors. Thus, $k > i$ and the set of predecessors of C_k in G' is included in $\{C_1, ..., C_{i-1}\}$. Since G' is a valid plan, the set of predecessors of C_k in $\mathcal{G}$ is included in $\{C_1, ..., C_{i-1}\}$. Then, we prove that the value L_{C_k} computed at line 9 of the algorithm at step i verifies $L_{C_k} \leq L_{G'}(C_k)$ (see Property A below). Then $L_G(C_i) \leq L_{C_k} \leq L_{G'}(C_k) \leq L_{G'}(C_i)$.
(ii) If the set of predecessors of C_i in G' is included in $\{C_1, ..., C_{i-1}\}$, then we can prove that $L_{G'}(C_i) \geq L_{C_i} = L_G(C_i)$, where L_{C_i} is the value computed at step i (see Property B below).
In both cases (i) and (ii), since $L_{G'}(C_i)$ is optimal, we have $L_G(C_i) = L_{G'}(C_i)$, thus proving the optimality of $L_G(C_i)$.

Proof of Properties A and B. Let C_k be a service with $k \geq i$ ($k > i$ for Property A, $k = i$ for Property B). Let G' be a valid plan such that the set of predecessors of C_k is included in $\{C_1, ..., C_{i-1}\}$. Then we prove that $L_{G'}(C_k) \geq L_{C_k}$, where L_{C_k} is the value computed at step i of the algorithm. Let $S = \{C_{u_1}, ..., C_{u_l}\}$ be the set of predecessors of C_k in G'. Let S' be the set of services that are either in S, or predecessor of a service of S in G. Let us show that $\prod_{C_i \in S} \sigma_i \geq \prod_{C_i \in S'} \sigma_i$. Let S_1 be the set of predecessors of C_{u_1} in G, S_2 the set of predecessors of C_{u_2} in G not in $S_1 \cup \{C_{u_1}\}$ and for all i S_i the set of predecessors of C_{u_i} in G not in $\bigcup_{j<i} S_j \cup \{C_{u_{i_1}}, ..., C_{u_{i-1}}\}$. Suppose that for one of the sets S_i, the product of selectivities $\prod_{C_j \in S_i} \sigma_j$ is strictly greater than one. Then $S_1 \cup ... \cup S_{i-1} \cup \{C_{u_{i_1}}, ..., C_{u_{i-1}}\}$ is a valid subset for C_{u_i} because G' is a valid plan and the product of selectivities on this subset is strictly smaller than the product of selectivities of the predecessors of C_{u_i} in G. This is in contradiction with the optimality of the set of predecessors of C_{u_i} chosen at line 7 of the algorithm. This proves that for all i, $\prod_{C_j \in S_i} \sigma_j \leq 1$. In addition, for all $j < i$, $L_G(C_j)$ is optimal. Hence the latency of C_k in G with S' as predecessor is smaller or equal to its latency in G', which proves that $L_{G'}(C_k) \geq L_{C_k}$.

Thus for $1 \leq i \leq n$, $L_G(C_i)$ is optimal, and therefore the plan computed by Algorithm 2 is optimal. □

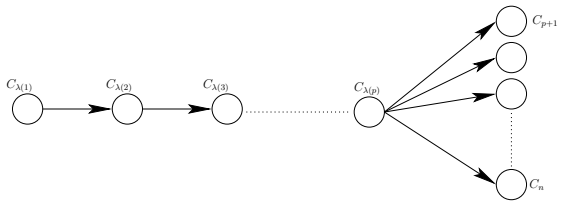

Figure 6. General structure for period minimization.

4.2 Bi-criteria problem

Theorem 3. *Problem* BICRITERIA-NOPREC-HOM *is polynomial and of complexity at most* $O(n^2)$. *Problem* BICRITERIA-PREC-HOM *is polynomial and of complexity at most* $O(n^6)$.

Proof. For each problem it is possible to modify the algorithm that minimizes the latency. See the extended version [4] for the proof. □

5 Heterogeneous platforms

In this section we investigate the optimization problems with heterogeneous resources. We show that both period and latency minimization problems are NP-complete, even for services without precedence constraints. Thus, bi-criteria problems on heterogeneous platforms are NP-complete. We also prove that there exists no approximation algorithm for MINPERIOD-NOPREC-HET with a constant factor, unless P=NP.

5.1 Period

In this section, we show that problem MINPERIOD-NOPREC-HET is NP-complete. The following property was presented in [12] for homogeneous platforms, and we extend it to different-speed servers.

Proposition 1. *Let* $(\mathcal{F}, \mathcal{S})$ *be an instance of the problem* MINPERIOD-NOPREC-HET. *We suppose* $\sigma_1, \sigma_2, ..., \sigma_p < 1$ *and* $\sigma_{p+1}, ..., \sigma_n \geq 1$. *Then the optimal period is obtained with a plan as in Figure 6.*

Proof. Let G be an optimal plan for this instance. We will not change the allocation of services to servers. Hence, in the following, C_i denotes the pair (C_i, S_u), with S_u the server assigned to C_i in G. Let $i, j \leq p$ (recall that p is the largest index of services whose selectivity is smaller than 1). Suppose that C_i is not an ancestor of C_j and that C_j is not an ancestor of C_i. Let $A'_k(G) = \mathsf{Ancest}_k(G) \cap \{C_1, ..., C_p\}$. Informally, the

idea is to add the arc (C_i, C_j) to G and to update the list of ancestors of each node (in particular, removing all nodes whose selectivity is greater than or equal to 1). Specifically, we construct the graph G' such that:

- for every $k \leq p$ such that $C_i \notin \mathsf{Ancest}_k(G)$ and $C_j \notin \mathsf{Ancest}_k(G)$, $\mathsf{Ancest}_k(G') = A'_k(G)$
- for every $k \leq p$ such that $C_i \in \mathsf{Ancest}_k(G)$ or $C_j \in \mathsf{Ancest}_k(G)$, $\mathsf{Ancest}_k(G') = A'_k(G) \cup A'_i(G) \cup A'_j(G) \cup \{C_i, C_j\}$
- $\mathsf{Ancest}_i(G') = A'_i(G)$
- $\mathsf{Ancest}_j(G') = A'_j(G) \cup A'_i(G) \cup \{C_i\}$
- for every $k > p$, $\mathsf{Ancest}_k(G') = \{C_1, ..., C_p\}$

In G', C_i is a predecessor of C_j and for all $p < k \leq n$, C_k has no successor. Also, because C_i and C_j were not linked by a precedence relation in G, G' is always a DAG (no cycle). In addition, for every node C_k of G, we have $\mathsf{Ancest}_k(G') \supset A'_k(G) = \mathsf{Ancest}_k(G) \cap \{C_1, ..., C_p\}$. This property implies:

$$\begin{array}{rcl} cost_k(G') & = & \frac{c_k}{s_u} \times \prod_{C_l \in \mathsf{Ancest}_k(G')} \sigma_l \\ & \leq & \frac{c_k}{s_u} \times \prod_{C_l \in A'_k(G)} \sigma_l \\ & \leq & \frac{c_k}{s_u} \times \prod_{C_l \in \mathsf{Ancest}_k(G)} \sigma_l \\ & \leq & cost_k(G). \end{array}$$

Hence, $\mathcal{P}(G') \leq \mathcal{P}(G)$ (recall that $\mathcal{P}(G)$ denotes the period of G). Because G was optimal, $\mathcal{P}(G') = \mathcal{P}(G)$, and G' is optimal too. By induction we construct a plan with the structure of Figure 6. □

We point out that only the *structure* of the plan is specified by Proposition 1. There remains to find the optimal ordering of services C_1 to C_p in the chain (this corresponds to the permutation λ in Figure 6), and to find the optimal assignment of services to servers.

Theorem 4. MINPERIOD-NOPREC-HET *is NP-complete.*

Proof. Consider the decision problem associated to MINPERIOD-NOPREC-HET: given an instance of the problem with n services and $p \geq n$ servers, and a bound K, is there a plan whose period does not exceed K? This problem obviously is in NP: given a bound and a mapping, it is easy to compute the period, and to check that it is valid, in polynomial time.

To establish the completeness, we use a reduction from RN3DM, a special instance of Numerical 3-Dimensional Matching that has been proved to be strongly NP-complete by Yu [18, 19]. Consider the following general instance $\mathcal{I}_1$ of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$ of size n, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{1}$$

We can suppose that $2 \leq A[i] \leq 2n$ for all i and that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance has no solution. Then we point out that Equation 1 is equivalent to

$$\begin{array}{rl} & \forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) \geq A[i] \\ \iff & \forall 1 \leq i \leq n, \quad \left(\frac{1}{2}\right)^{\lambda_1(i)-1} \times \frac{2^{A[i]}}{2^{\lambda_2(i)}} \leq 2 \end{array} \tag{2}$$

We build the following instance $\mathcal{I}_2$ of MINPERIOD-HET with n services and $p = n$ servers such that $c_i = 2^{A[i]}$, $\sigma_i = 1/2$, $s_i = 2^i$ and $K = 2$. The size of instance $\mathcal{I}_1$ is $O(n \log(n))$, because each $A[i]$ is bounded by $2n$. In fact, because RN3DM is NP-complete in the strong sense, we could encode $\mathcal{I}_1$ in unary, with a size $O(n^2)$, this does not change the analysis. We encode the instance of $\mathcal{I}_1$ with a total size $O(n^2)$, because the c_i and s_i have size at most $O(2^n)$, hence can be encoded with $O(n)$ bits each, and there are $O(n)$ of them. The size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$.

Now we show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Assume first that $\mathcal{I}_1$ has a solution. Then we build a plan which is a linear chain. Service C_i is at position $\lambda_1(i)$, hence is filtered $\lambda_1(i) - 1$ times by previous services, and it is processed by server $S_{\lambda_2(i)}$, matching the cost in Equation 2.

Reciprocally, if we have a solution to $\mathcal{I}_2$, then there exists a linear chain G with period 2. Let $\lambda_1(i)$ be the position of service C_i in the chain, and let $\lambda_2(i)$ be the index of its server. Equation 2 is satisfied for all i, hence Equation 1 is also satisfied for all i: we have found a solution to $\mathcal{I}_1$. This completes the proof.

We add the following observation which we will need below: the only solutions are chains. Indeed, if we had a solution G' that was not a chain, we would transform it step by step into a chain, according to Proposition 1. At each step of transformation, we consider a pair (C, C') and we add an edge $C \to C'$. If at a given step we add the edge $C_i \to C_j$, we have $\mathsf{Ancest}_j(G') \subsetneq \mathsf{Ancest}_j(G)$. However $cost_j(G) = 2$ and all the selectivities are strictly lower than 1. Then $cost_j(G') > 2$. That contradicts the hypothesis of optimality of G'. This proves the claim. □

The proof also shows that the problem remains NP-complete when all service selectivities are identical.

Proposition 2. *For any $K > 0$, there exists no K-approximation algorithm for* MINPERIOD-NOPREC-HET*, unless P=NP.*

Proof. Suppose that there exists a polynomial algorithm that computes a K-approximation of this problem. We use the same instance $\mathcal{I}_1$ of RN3DM as in the proof of Theorem 4.

Let $\mathcal{I}_2$ be the instance of our problem with n services such that, for $1 \leq i \leq n$, $c_i = (2K)^{A[i]-1}$, $\sigma_i = \frac{1}{2K}$, $s_i = (2K)^i$ and $P = 1$. From the previous proof, we know that the only solutions are chains. In such a solution chain, service C_i is placed in position $\lambda_1(i)$ in the chain, and it is processed by server $S_{\lambda_2(i)}$, where (λ_1, λ_2) is a solution of $\mathcal{I}_1$. In such solutions, we obtain $cost_i(G) = 2$ for each i.

In any other solution, there is a service whose computation cost is larger than $P = 1$. In addition, all computation costs are integer power of $2K$. That means that in any other solution, the period is greater or equal to $2K$. Hence the only K-approximations are the optimal solutions. If a polynomial algorithm finds such a solution, we can compute the permutations λ_1 and λ_2 and solve $\mathcal{I}_1$ in polynomial time. This contradicts the hypothesis $P \neq NP$. □

5.2 Latency

We first show that the optimal solution of MINLATENCY-NOPREC-HET has a particular structure. We then use this result to derive the NP-completeness of the problem.

Definition 1. *Given a plan G and a vertex $v = (C_i, S_u)$ of G, (i) v is a leaf if it has no successor in G; and (ii) $d_i(G)$ is the maximum length (number of links) in a path from v to a leaf. If v is a leaf, then $d_i(G) = 0$.*

Proposition 3. *Let $C_1, ..., C_n, S_1, ..., S_n$ be an instance of* MINLATENCY*. Then, the optimal latency can be obtained with a plan G such that, for any couple of nodes of G $v_1 = (C_{i_1}, S_{u_1})$ and $v_2 = (C_{i_2}, S_{u_2})$,*

1. *If $d_{i_1}(G) = d_{i_2}(G)$, v_1 and v_2 have the same predecessors and the same successors in G.*
2. *If $d_{i_1}(G) > d_{i_2}(G)$ and $\sigma_{i_2} \leq 1$, then $c_{i_1}/s_{u_1} < c_{i_2}/s_{u_2}$.*
3. *All nodes with a service of selectivity $\sigma_i > 1$ are leaves ($d_i(G) = 0$).*

Proof. Let G be an optimal plan for this instance. We will not change the allocation of services to servers, so we can design vertices of the graph as C_i only, instead of (C_i, S_u). We want to produce a graph G' which verifies Proposition 3.

Property 1. In order to prove Property 1 of the proposition, we recursively transform the graph G, starting from the leaves, so that at each level every nodes have the same predecessors and successors.

For every vertex C_i of G, we recall that $d_i(G)$ is the maximum length of a path from C_i to a leaf in G. Let $A_i = \{C_j |\ d_j(G) = i\}$. A_0 is the set of the leaves of G. We denote by G_i the subgraph $A_0 \cup ... \cup A_i$. Note that these subgraphs may change at each step of the transformation, and they are always computed with the current graph G.

• *Step* 0. Let $c_i = \max_{C_j \in A_0} c_j$. Let G' be the plan obtained from G by changing the predecessors of every service in A_0 such that the predecessors of a service of A_0 in G' are exactly the predecessors of C_i in G. Let B_i be the set of predecessors of C_i in G and let $C_j \in B_i$ be the predecessor of C_i of maximal completion time. The completion time of a service C_ℓ of $G - A_0$ does not change: $L_{G'}(C_\ell) = L_G(C_\ell)$. And, for each service C_k in A_0,

$$\begin{aligned} L_{G'}(C_k) &= L_{G'}(C_j) + \left(\prod_{C_\ell \in B_i} \sigma_\ell\right) \times c_k \\ &\leq L_{G'}(C_j) + \left(\prod_{C_\ell \in B_i} \sigma_\ell\right) \times c_i \\ &\leq L_{G'}(C_i) = L_G(C_i) \end{aligned}$$

Therefore, $\forall C_k \in A_0,\ L_{G'}(C_k) \leq L_G(C_i)$. Since for $C_k \notin A_0, L_{G'}(C_k) \leq L_G(C_k)$, and since G was optimal for the latency, we deduce that G' is also optimal for the latency. This completes the first step of the modification of the plan G.

• *Step i.* Let i be the largest integer such that G_i verifies Property 1. If $G_i = G$, we are done since the whole graph verifies the property. Let $C_{i'}$ be a node such that $L_{G_i}(C_{i'}) = \max_k L_{G_i}(C_k)$. Note that these finish times are computed in the subgraph G_i, and thus they do not account for the previous selectivities in the whole graph G. Let C_j be an entry node of G_i (no predecessors in G_i) in a path realizing the maximum time $L_{G_i}(C_{i'})$, and let C_ℓ be the predecessor in G of C_j of maximal finish time $L_G(C_\ell)$. Then G' is the plan obtained from G in changing the predecessors of every service of A_i such that the predecessors of a service of A_i in G' are the predecessors of C_j in G. For $C_k \in G \setminus G_i$, we have $L_{G'}(C_k) = L_G(C_k)$. Let C_k be a node of G_i. We have:

$$\begin{aligned} L_{G'}(C_k) &= L_{G'}(C_\ell) + \left(\prod_{C_m \in \mathsf{Ancest}_j(G')} \sigma_m\right) \times L_{G_i}(C_k) \\ &\leq L_G(C_\ell) + \left(\prod_{C_m \in \mathsf{Ancest}_j(G)} \sigma_m\right) \times L_{G_i}(C_{i'}) \\ &\leq L(G) \end{aligned}$$

and $L(G)$ is optimal. So, $L(G') = L(G)$.

• *Termination of the algorithm.* Let C_k be a node of G. If C_k is a predecessor of C_j in G or if $C_k \in G_i$, then $d_k(G') = d_k(G)$. Otherwise, every path from C_k to a leaf in G has been removed in G', so $d_k(G') < d_k(G)$. This proves that $\sum_j d_j(G) \geq \sum_j d_j(G')$.

- If, at the end of step i, $\sum_j d_j(G) = \sum_j d_j(G')$, then G_{i+1} verifies Property 1, and we can go to step $i+1$.

- However, if $\sum_j d_j(G) > \sum_j d_j(G')$, some leaves may appear since we have removed successors of some nodes in the graph. In this case, we start again at step 0.

The algorithm will end because at each step, either the value $\sum_j d_j(G)$ decreases strictly, or it is equal but i increases. It finishes either if there are only leaves left in the graph at a step with $i = 0$, or when we have already transformed all levels of the graph and $G_i = G$.

Property 2. Let G be an optimal graph for latency verifying Property 1. Suppose that there exists a pair (C_i, S_u) and (C_j, S_v) such that $d_i(G) > d_j(G)$, $\sigma_J \leq 1$, and $c_i/s_u > c_j/s_v$. Let G' be the graph obtained by removing all the edges beginning and ending by (C_j, S_v) and by choosing as predecessors of (C_j, S_v) the predecessors of (C_i, S_u) in G and as successors of C_j the successors of C_i in G. Since $\sigma_j \leq 1$, the cost of successors can only decrease. The other edges do not change. $L(G') \leq L(G)$ and G is optimal, so G' is optimal and Property 1 of Proposition 3 is verified. We can continue this operation until Property 2 is verified.

Property 3. The last property just states that all nodes of selectivity greater than 1 are leaves. In fact, if such a node C_i is not a leaf in G, we remove all edges from C_i to its successors in the new graph G', thus only potentially decreasing the finish time of its successor nodes. Indeed, a successor will be able to start earlier and it will have less data to process. □

Lemma 1. *Let $C_1, ..., C_n, S_1, ..., S_n$ be an instance of* MINLATENCY-HET *such that for all i, c_i and s_i are integer power of 2 and $\sigma_i \leq \frac{1}{2}$. Then the optimal latency is obtained with a plan G such that*

1. *Proposition 3 is verified;*
2. *for all nodes (C_{i_1}, S_{u_1}) and (C_{i_2}, S_{u_2}) with $d_{i_1}(G) = d_{i_2}(G)$, we have $\frac{c_{i_1}}{s_{u_1}} = \frac{c_{i_2}}{s_{u_2}}$.*

Proof. Let G be a plan verifying Proposition 3. Suppose that there exists a distance to leaves d such that the nodes at this distance do not respect Property 2 of Lemma 1. Let A be the set of nodes (C_i, S_u) of maximal ratio $\frac{c_i}{s_u} = c$ with $d_i(G) = d$ and A' be the set of other nodes at distance d. Let c' be the maximal ratio $\frac{c_j}{s_v}$ of nodes $(C_j, S_v) \in A'$. Since $c' < c$ and c, c' are integer power of 2, we have $c' \leq \frac{c}{2}$.

We construct the plan G' such that:

- For all node $(C_i, S_u) \notin A$, $\mathsf{Ancest}_i(G') = \mathsf{Ancest}_i(G)$
- For all node $(C_i, S_u) \in A$, $\mathsf{Ancest}_i(G') = \mathsf{Ancest}_i(G) \cup A'$

The completion time of nodes of A' and of nodes of distance strictly greater than d in G does not change. Let T_d be the completion time of the service (C_k, S_v) at distance $d+1$ of maximal ratio $\frac{c_k}{s_v}$. Let (C_i, S_u) be a pair of A. Let $\sigma = \sum_{C_j \in \mathsf{Ancest}_i(G)} \sigma_j$. Then we have

$$\begin{aligned} T_i(G') &= T_d + \sigma \times c' + \sigma \times (\textstyle\sum_{C_j \in A'} \sigma_j) \times c \\ &\leq T_d + \sigma \times \tfrac{c}{2} + \sigma \times \tfrac{1}{2} \times c \\ &\leq T_d + \sigma \times c \\ &\leq T_i(G) \end{aligned}$$

This proves that the completion time of the services of A does not increase between G and G'. The completion time of services of distance smaller than d does not increase because their sets of predecessors do not change. G is a graph corresponding to Proposition 3, that means it obtains the optimal latency and the latency of G' is smaller or equal to the latency of G. We can conclude that G' is optimal for latency.

We obtain by this transformation an optimal plan G' for latency with strictly less pairs of nodes that does not correspond to the property of Lemma 1 than in G. In addition, G' respect properties of Proposition 3. By induction, we can obtain a graph as described in Lemma 1. □

Theorem 5. MINLATENCY-NOPREC-HET *is NP-complete.*

Proof. Consider the decision problem associated to MINLATENCY-HET: given an instance of the problem with n services and $p \geq n$ servers, and a bound K, is there a plan whose latency does not exceed K? This problem obviously is in NP: given a bound and a mapping, it is easy to compute the latency, and to check that it is valid, in polynomial time.

To establish the completness, we use a reduction from RN3DM. Consider the following general instance $\mathcal{I}_1$ of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$

of size n, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{3}$$

We can suppose that $\sum_{i=1}^{n} A[i] = n(n+1)$. We build the following instance $\mathcal{I}_2$ of MINLATENCY-HET such that:

- $c_i = 2^{A[i] \times n + (i-1)}$
- $\sigma_i = \left(\frac{1}{2}\right)^n$
- $s_i = 2^{n \times (i+1)}$
- $K = 2^n - 1$

The size of instance $\mathcal{I}_1$ is $O(nlog(n))$, because each $A[i]$ is bounded by $2n$. The new instance $\mathcal{I}_2$ has size $O(n \times (n^2))$, since all parameters are encoded in binary. The size of $\mathcal{I}_2$ is thus polynomial in the size of $\mathcal{I}_1$.

Now we show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution.

Suppose first that $\mathcal{I}_1$ has a solution λ_1, λ_2. We place the services and the servers on a chain with service C_i on server $S_{\lambda_1(i)}$ in position $\lambda_2(i)$ on the chain. We obtain the latency

$$\begin{array}{rcl} L(G) & = & \sum_i \frac{c_i}{s_{\lambda_1(i)}} * \left(\frac{1}{2^n}\right)^{\lambda_2(i)-1} \\ & = & \sum_i 2^{A[i] \times n + (i-1) - n \times (\lambda_1(i)+1) - n \times (\lambda_2(i)-1)} \\ & = & \sum_i 2^{(A[i] - \lambda_1(i) - \lambda_2(i)) \times n + (i-1)} \\ & = & \sum_{i=1}^{n} 2^{i-1} \\ & = & 2^n - 1 \end{array}$$

This proves that if $\mathcal{I}_1$ has a solution then $\mathcal{I}_2$ has a solution.

Suppose now that $\mathcal{I}_2$ has a solution. Let G be an optimal plan that respects properties of Lemma 1. Let (C_{i_1}, S_{u_1}), (C_{i_2}, S_{u_2}) be two distinct nodes of G. Let a_1 and a_2 be two integers such that $\frac{c_{i_1}}{s_{u_1}} = 2^{a_1}$ and $\frac{c_{i_2}}{s_{u_2}} = 2^{a_2}$. The rest of the Euclidean division of a_1 by n is equal to $i_1 - 1$, and the rest of the Euclidean division of a_2 by n is equal to $i_2 - 1$. Since both nodes are distinct, $i_1 \neq i_2$ and we can conclude that $\frac{c_{i_1}}{s_{u_1}} \neq \frac{c_{i_2}}{s_{u_2}}$. The ratios cost/speed are all different and G verifies properties of Lemma 1. As a result, G is a linear chain.

Let λ_1, λ_2 be two permutations such that for all i, the service C_i is in position $\lambda_2(i)$ on the server $S_{\lambda_1(i)}$. We want to achieve a latency strictly smaller than 2^n, and thus for every node $(C_i, S_{\lambda_1}(i))$,

$$\begin{array}{rr} & 2^{A[i] \times n + (i-1) - n \times (\lambda_1(i)+1) - n \times (\lambda_2(i)-1)} < 2^n \\ \iff & 2^{(A[i] - \lambda_1(i) - \lambda_2(i)) \times n + (i-1)} < 2^n \\ \iff & A[i] - \lambda_1(i) - \lambda_2(i) \leq 0 \end{array}$$

This proves that λ_1, λ_2 is a valid solution of $\mathcal{I}_1$. Thus, $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution, which concludes the proof. □

Proposition 4. *For any $K > 0$, there exists no K-approximation algorithm for* MINLATENCY-NOPREC-HET*, unless P=NP.*

The proof is similar to that of Proposition 2. See the extended version [4] for full details.

5.3 Particular instances

In this section, we report the complexity of some particular instances of MINLATENCY-NOPREC-HET. Please refer to the extended version [4] for proofs:

• The problem remains NP-complete when we when we impose that the plan is a linear chain.

• The problem becomes polynomial when all services have same cost.

• The problem becomes polynomial when the optimal plan is a star, or a bipartite graph.

6 Conclusion

In this paper, we have considered the important problem of mapping filtering service applications onto computational platforms. Our main focus was to give an insight of the combinatorial nature of the problem, and to assess the impact of using heterogeneous resources on the problem complexity. We considered the two major objective functions, minimizing the period and minimizing the latency, and also studied bi-criteria optimization problems. Several instances of the problem have been shown NP-complete, while others can be solved with complex polynomial algorithms, such as the optimal algorithm for MINLATENCY-PREC-HOM. We believe that this exhaustive study will provide a solid theoretical foundation for the study of single criterion or bi-criteria mappings.

For NOPREC-HET, all problems (period, latency, and hence bi-criteria) are NP-complete. In the extended version [4], we provide an integer linear program and many heuristics for MINPERIOD-NOPREC-HET, and experiments show that in many cases our heuristics are close to the optimal solution returned by the linear program. We also have derived an integer linear program for MINLATENCY-NOPREC-HET, but its exponential number of variables renders it untractable, even for small problem instances.

As future work, we intend to design heuristics for the general problems MINPERIOD-PREC-HET and MINLATENCY-PREC-HET, and to derive lower bounds so as to assess their performance. Also, extending ideas of task graph replication algorithms (such as those in [2]) to the framework of pipelined workflows with filtering services looks a promising direction to further explore.

Acknowledgment

We thank the reviewers for their comments and suggestions. This work was supported in part by the ANR StochaGrid project.

References

[1] A. Agnetis, P. Detti, M. Pranzo, and M. S. Sodhi. Sequencing unreliable jobs on parallel machines. *Journal on Scheduling*, 2008. Available on-line at `http://www.springerlink.com/content/c571u1221560j432`.

[2] I. Ahmad and Y.-K. Kwok. On exploiting task duplication in parallel program scheduling. *IEEE Trans. Parallel Distributed Systems*, 9(9):872–892, 1998.

[3] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom. Adaptive ordering of pipelined stream filters. In *SIGMOD'04: Proceedings of the 2004 ACM SIGMOD Int. Conf. on Management of Data*, pages 407–418. ACM Press, 2004.

[4] A. Benoit, F. Dufossé, and Y. Robert. On the complexity of mapping filtering services on heterogeneous platforms. Research Report 2008-30, LIP, ENS Lyon, France, Oct. 2008. Available at `graal.ens-lyon.fr/~fdufosse/`.

[5] A. Benoit and Y. Robert. Mapping pipeline skeletons onto heterogeneous platforms. *J. Parallel Distributed Computing*, 68(6):790–808, 2008.

[6] J. Burge, K. Munagala, and U. Srivastava. Ordering pipelined query operators with precedence constraints. Research Report 2005-40, Stanford University, November 2005.

[7] S. Chaudhuri and K. Shim. Optimization of queries with user-defined predicates. *ACM Trans. Database Systems*, 24(2):177–228, 1999.

[8] DataCutter Project: Middleware for Filtering Large Archival Scientific Datasets in a Grid Environment. `http://www.cs.umd.edu/projects/hpsl/ResearchAreas/DataCutter.htm`.

[9] D. Florescu, A. Grunhagen, and D. Kossmann. Xl: A platform for web services. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research*, 2003. On-line proceedings at `http://www-db.cs.wisc.edu/cidr/program/p8.pdf`.

[10] J. M. Hellerstein. Predicate migration: Optimizing queries with expensive predicates. In *In Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 267–276, 1993.

[11] M. Ouzzani and A. Bouguettaya. Query processing and optimization on the web. *Distributed and Parallel Databases*, 15(3):187–218, 2004.

[12] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query optimization over web services. In *VLDB '06: Proceedings of the 32nd Int. Conference on Very Large Data Bases*, pages 355–366. VLDB Endowment, 2006.

[13] K. Taura and A. A. Chien. A heuristic algorithm for mapping communicating tasks on heterogeneous resources. In *Heterogeneous Computing Workshop*, pages 102–115. IEEE Computer Society Press, 2000.

[14] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz. An approach for optimizing latency under throughput constraints for application workflows on clusters. Research Report OSU-CISRC-1/07-TR03, Ohio State University, Columbus, OH, Jan. 2007. Available at `ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007.ShortversionappearsinEuroPar'2008`.

[15] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz. Optimizing latency and throughput of application workflows on clusters. Research Report OSU-CISRC-4/08-TR17, Ohio State University, Columbus, OH, Apr. 2008. Available at `ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007`.

[16] Q. Wu, J. Gao, M. Zhu, N. Rao, J. Huang, and S. Iyengar. On optimal resource utilization for distributed remote visualization. *IEEE Trans. Computers*, 57(1):55–68, 2008.

[17] Q. Wu and Y. Gu. Supporting distributed application workflows in heterogeneous computing environments. In *14th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE Computer Society Press, 2008.

[18] W. Yu. *The two-machine flow shop problem with delays and the one-machine total tardiness problem*. PhD thesis, Technishe Universiteit Eidhoven, June 1996.

[19] W. Yu, H. Hoogeveen, and J. K. Lenstra. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *J. Scheduling*, 7(5):333–348, 2004.