Periodicity is Optimal for Offline and Online Multi-Stage Adjoint Computations

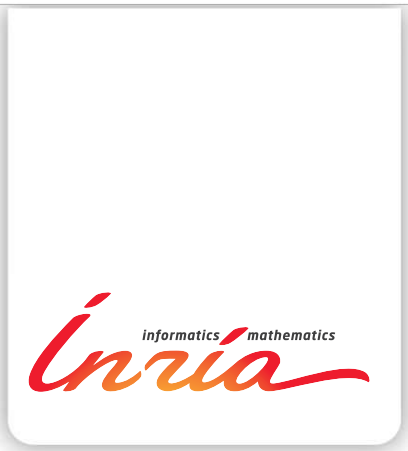
Guillaume Aupy, Julien Herrmann

Periodicity is Optimal for Offline and Online Multi-Stage Adjoint Computations

Guillaume Aupy, Julien Herrmann

RESEARCH REPORT
N° 8822
December 2015
Project-Team ROMA

ISSN 0249-6399 ISRN INRIA/RR--8822--FR+ENG



Periodicity is Optimal for Offline and Online Multi-Stage Adjoint Computations

Guillaume Aupy*, Julien Herrmann†

Project-Team ROMA

Research Report n° 8822 — December 2015 — 36 pages

Abstract: We reexamine the work of Aupy et al. on optimal algorithms for multi-stage adjoint computations, where two levels of memories are available. Previous optimal algorithm had a quadratic execution time. Here, with structural arguments, namely periodicity, on the optimal solution, we provide an optimal algorithm in constant time, with appropriate pre-processing. We also provide an asymptotically optimal algorithm for the online problem, when the adjoint graph size is not known before-hand. Again, this algorithms rely on the proof that the optimal solution for multi-stage adjoint computations is weakly periodic. We conjecture a closed-form formula for the period. Finally, we experimentally access the convergence speed of the approximation ratio for the online problem, through simulations.

Key-words: Optimal algorithms; Out-of-core; Adjoint Computation; Program Reversal; Automatic Differentiation; Online Adjoint Computation; Periodicity.

* Penn State University, Pennsylvanie, USA
† LIP, École Normale Supérieure de Lyon, INRIA, France

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Algorithmes à plusieurs étages optimaux pour le calcul d'adjoints en ligne et hors-ligne

Résumé : Dans cet article, nous améliorons le travail de Aupy et al. sur les algorithmes à plusieurs étages optimaux pour le calcul d'adjoints, avec deux niveaux de mémoire disponibles. L'algorithme optimal précédent avait un temps d'exécution quadratique. Ici, avec une analyse de la solution optimale, nous proposons un algorithme optimal en temps constant, avec précalculs. Nous proposons également un algorithme asymptotiquement optimal pour la version en ligne du problème, lorsque la taille du graphe adjoint n'est pas connue à l'avance. Ces algorithmes reposent sur la preuve que les solutions optimales pour le calcul d'adjoint sont périodiques. Nous conjecturons la formule close de cette période. Enfin, nous évaluons la vitesse de convergence du ratio d'approximation pour le problème en ligne, à travers une campagne de simulations.

Mots-clés : Algorithmes optimaux; out-of-core; calcul d'adjoints; retournement de programmes; différentiation automatique; calcul d'adjoints en ligne; périodicité.

Contents

1 Introduction **4**

2 Framework **5**
2.1 The AC problem 5
2.2 Summary of our previous work 7

3 Dominant Sequences and Dominant Optimal Sequences **9**

4 Main contributions **11**

5 Showing the periodicity **12**
5.1 Properties of a period 12
5.1.1 Properties of $\mathrm{Opt}_0(l, c_m)$ 12
5.1.2 Properties of $\mathrm{Opt}_1^{(d)}(l, c_m, r_d)$ 14
5.2 Execution time of a DS 17
5.3 Existence of a DOS 19
5.4 Admissible periods $\mathcal{M}_X^{(\mathrm{Adm})}$ 20
5.5 Bounding the size of a period 22
5.5.1 Maximum period size 22
5.6 Construction of $\mathcal{M}_X$ and periodicity 23
5.6.1 Bounding the number of solutions with less than one period 23
5.6.2 M-DOS and periodicity 26
5.7 Experimental evaluation of i_X and m_X 30
5.7.1 Can we get a better bound for i_X? 30
5.7.2 Can we compute m_X? 30

6 Asymptotically optimal online algorithm **31**

7 Evaluation of a periodic schedule **33**
7.1 Periodic Dominant Schedules 33
7.2 Experimental Results 34

8 Conclusion **35**

1 Introduction

The need to efficiently compute the gradient to a cost function arises frequently in many areas of scientific computing, including mathematical optimization, uncertainty quantification, and nonlinear systems of equations. In gradient based optimization, discrete adjoint methods are widely used for the gradient computation when the problem possesses an extensive amount of design variables. The initial differential equation is discretised, and the discrete adjoint equations immediately follow. The derivative computation applies the chain rule of differential calculus starting with the dependent variables and propagating back to the independent variables. Thus, it reverses the flow of the original function evaluation. In the case of linear governing equations, only the final solution data from the forward solve is required, however, in the case of nonlinear governing equations the solution data must be stored at every time step which can become restrictive for large problems. In general, intermediate function values must be stored or recomputed [2].

A popular storage or recomputation strategy for functions that have some sort of natural "time step" is to save (*checkpoint*) the state at each time step during the function computation (*forward sweep*) and use this saved state in the derivative computation (*reverse sweep*). If the storage is inadequate for all states, one can checkpoint only some states and recompute the unsaved states as needed. If the number of time step is know a priori, Griewank and Walther proved that, given a fixed number of checkpoints, the schedule that minimize the amount of recomputation is a binomial checkpointing strategy [3, 4]. They also gave a closed formula to compute the indices of the forward step to store in the memory [10]. The problem model they used implicitly assumes that reading and writing checkpoints are free, but the number of available checkpoint is limited (*one memory framework*). In [8], Stumm and Walther consider the case where another type of storage is available. The checkpoints can be written onto a disk with an unlimited storage capacity but whose time to read or write a checkpoint can no longer be ignored (*multi-stage framework*). They designed a multi-stage heuristic using a binomial checkpointing strategy based on the optimal algorithm for the case without disk. More recently, Aupy et al. [1] provided a polynomial time algorithm for determining the optimal schedule using both memory and disk, if the number of time step is know a priori (*offline framework*).

However, in the context of flow control, the partial differential equations to be solved are usually stiff, and the solution process relies therefore on some adaptive time stepping procedure. Hence, the number of time steps performed is known only after the complete integration (*online framework*). One has to decide on the fly during the forward sweep where to place the checkpoints without knowing how many time steps are left to perform. In [5], Heuveline and Walter provided an algorithm that computes the time-minimal schedule for an unknown number of time steps as long as this number does not exceed a given upper bound. This heuristic is designed for the *one memory framework* and relies on a dynamic rearrangement of the checkpoints. Later, Stumm and Walther [9] designed a new algorithm to compute almost optimal online check-

pointing schedules as long as the number of time steps does not exceed a larger than before upper bound. For larger adjoint computations, Wang and al. [11] provided a dynamic online algorithm that minimizes the maximum number of recomputation for a single forward step, matching the optimal offline repetition number and ensuring that the overall computational cost has a theoretical upper bound. To the best of our knowledge, there is no previous study on online multi-stage adjoint computations, since no optimal algorithm for the offline multi-stage framework were known before [1] bridged that gap. This paper aims to extend their results to the online framework.

In this paper, we provide a theoretical analysis of the optimal offline multi-stage algorithm proposed in [1], which allows us to reduced significantly its computation time. This optimization relies on the proof that for any adjoint graph size, there exists an optimal disk checkpointing strategy that is weakly periodic, which means that all the intervals between two consecutive disk checkpoints are of the same size except for a bounded number of them. This bound and the optimal interval only depend on the architecture parameters, namely the number of available memory checkpoints and the time to read or write a checkpoint on the disk. This observations allow us to provide two valuable contributions. First, we can design an optimal algorithm in constant time (independent of the adjoint graph size) for the offline multi-stage framework, with an appropriate pre-processing depending on the architecture parameters. Second, we can design an asymptotically optimal algorithm for the online multi-stage framework.

The rest of this paper is organized as follows. Section 2 lays the grounds of multi-stage adjoint computation and recalls important results from [1]. Section 3 introduces notations that will be used during the paper. In Section 4 we introduce the main results of this paper. Section 5 constitutes the hearth of this paper and provide many structural arguments on the optimal multi-stage schedules. This results are used in Section 6 to design the asymptotically optimal algorithm for the online multi-stage framework. Finally, Section 7 access the performance of the asymptotically optimal online algorithm compared to the actual optimal algorithm that can be computed when the actual size of the adjoint graph is known beforehand.

2 Framework

2.1 The AC problem

Definition 1 (Adjoint Computation (AC) [4, 8])**.** An adjoint computation (AC) with l time steps can be described by the following set of equations:

$$F_i(x_i) = x_{i+1} \text{ for } 1 \leq i < l \tag{1}$$

$$\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \text{ for } 1 \leq i \leq l \tag{2}$$

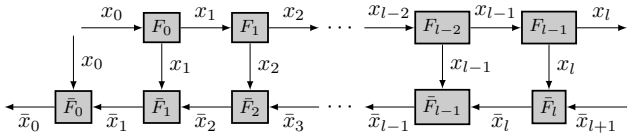


Figure 1: The AC dependence graph.

The dependencies between these operations[1] are represented by the graph $\mathcal{G} = (V, E)$ depicted in Figure 1.

The F computations are called *forward* steps. They have an execution cost of u_f. The $\bar{F}$ computations are called *backward* steps, they have an execution cost of u_b. If $\bar{x}_l$ is initialized appropriately, then at the conclusion of the adjoint computation, $\bar{x}_0$ will contain the gradient with respect to the initial state (x_0).

Definition 2 (Platform)**.** We consider a platform with three storage locations:

- *Buffers:* there are two buffers, the top buffer and the bottom buffer. The top buffer is used to store a value x_i for some i, while the bottom buffer is used to store a value $\bar{x}_i$ for some i. For a computation (F or $\bar{F}$) to be executed, its input values have to be stored in the buffers. Let $\mathcal{B}^\top$ and $\mathcal{B}^\bot$ denote the content of the top and bottom buffers. In order to start the execution of the graph, x_0 must be stored in the top buffer, and $\bar{x}_{l+1}$ in the bottom buffer. Hence without loss of generality, we assume that at the beginning of the execution, $\mathcal{B}^\top = \{x_0\}$ and $\mathcal{B}^\bot = \{\bar{x}_{l+1}\}$.

- *Memory:* there are c_m slots of memory where the content of a buffer can be stored. The time to write from buffer to memory is w_m. The time to read from memory to buffer is r_m. Let $\mathcal{M}$ be the set of x_i and $\bar{x}_i$ values stored in the memory. The memory is empty at the beginning of the execution ($\mathcal{M} = \emptyset$).

- *Disks:* there are c_d slots of disks where the content of a buffer can be stored.. The time to write from buffer to disk is w_d. The time to read from disk to buffer is r_d. Let $\mathcal{D}$ be the set of x_i and $\bar{x}_i$ values stored in the disk. The disk is empty at the beginning of the execution ($\mathcal{D} = \emptyset$).

Memory and disk are generic terms for a two-level storage system, modeling any platform with a dual memory system, including (i) a cheap-to-access first-level memory, of limited size; and (ii) and a costly-to-access second-level memory, whose size is very large in comparison of the first-level memory. The pair *(memory, disk)* can be replaced by *(cache, memory)* or *(disk, tape)* or any relevant hardware combination.

[1] In the original approach by Griewank [3], an extra F_l operation was included. It is not difficult to take this extra operation into account.

Intuitively, the core of the AC problem is the following: after the execution of a forward step, its output is kept in the top buffer only. If it is not saved in memory or disk before the next forward step, it is lost and will have to be recomputed when needed for the corresponding backward step. When no disk storage is available, the problem is to minimize the number of re-computations in the presence of limited (but free-to-access) memory slots. When disk storage is added, the problem becomes even more challenging: saving data on disk can save some recompilation, and a trade-off must be found between the cost of disk accesses and that of recomputations.

Here, we consider the problem with an unlimited number of disk storage, where, while reading and writing from/to memory is still free, reading and writing from/to disk has a cost. We assume that at the beginning of the execution, both the memory and the disk are empty:

Problem 1 ($\textsc{Prob}_\infty(l, c_m, w_d, r_d)$)**.** We want to minimize the makespan of the AC problem with the following parameters:

		Initial state:
AC graph:	size l	
Steps:	u_f, u_b	
Memory:	c_m, $w_m = r_m = 0$	$\mathcal{M}_{\text{ini}} = \emptyset$
Disks:	$c_d = +\infty, w_d, r_d$	$\mathcal{D}_{\text{ini}} = \emptyset$
Buffers:	$\mathcal{B}^\top$, $\mathcal{B}^\perp$	$\mathcal{B}^\top_{\text{ini}} = \{x_0\}$, $\mathcal{B}^\perp_{\text{ini}} = \{\bar{x}_{l+1}\}$

Definition 3 ($\text{Opt}_\infty(l, c_m, w_d, r_d)$)**.**
Given $l \in \mathbb{N}$, $c_m \in \mathbb{N}$, $w_d \in \mathbb{R}$ and $r_d \in \mathbb{R}$, $\text{Opt}_\infty(l, c_m, w_d, r_d)$ is the execution time of an optimal solution to $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$.

2.2 Summary of our previous work

In our previous work [1], we were able to compute in polynomial time an optimal solution to $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$: Disk-Revolve. A solution to $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ is a sequence of operations from Table 1 that executes an AC graph of size l in minimal time.

In our previous work [1], we solved $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ by reducing it to another problem: $\overline{\textsc{Prob}}_1^{(d)}(l, c_m, r_d)$ (see Problem 2). We defined $\text{Opt}_1^{(d)}(l, c_m, r_d)$ as the execution time of a solution to $\overline{\textsc{Prob}}_1^{(d)}(l, c_m, r_d)$.

Problem 2 ($\overline{\textsc{Prob}}_1^{(d)}(l, c_m, r_d)$)**.** We want to minimize the makespan of the AC problem with the following parameters.

		Initial state:
AC graph:	size l	
Steps:	u_f, u_b	
Memory:	c_m, $w_m = r_m = 0$	$\mathcal{M}_{\text{ini}} = \emptyset$
Disks:	$c_d = 1, w_d = +\infty, r_d$	$\mathcal{D}_{\text{ini}} = \{x_0\}$
Buffers:	$\mathcal{B}^\top$, $\mathcal{B}^\perp$	$\mathcal{B}^\top_{\text{ini}} = \{x_0\}$, $\mathcal{B}^\perp_{\text{ini}} = \{\bar{x}_{l+1}\}$

[2] Assuming $|\mathcal{M}| < c_m$, or $x_i \in \mathcal{M}$.

	Operation	Input	Action
F_i	Executes one forward computation F_i (for $i \in \{0,\dots,l-1\}$). This operation takes a time $\text{cost}(F_i) = u_f$.	$\mathcal{B}^\top = \{x_i\}$	$\mathcal{B}^\top \leftarrow \{x_{i+1}\}$
$F_{i\to i'}$	Denotes the sequence $F_i \cdot F_{i+1} \cdot \ldots \cdot F_{i'}$.	$\mathcal{B}^\top = \{x_i\}$	$\mathcal{B}^\top \leftarrow \{x_{i'}\}$
$\bar{F}_i$	Executes the backward computation $\bar{F}_i$ ($i \in \{0,\dots,l\}$). This operation takes a time $\text{cost}(\bar{F}_i) = u_b$.	$\mathcal{B}^\top = \{x_i\}$, $\mathcal{B}^\perp = \{\bar{x}_{i+1}\}$	$\mathcal{B}^\perp \leftarrow \{\bar{x}_i\}$
W_i^m	Writes the value x_i of the top buffer into the memory. This operation takes time $\text{cost}(W_i^m) = w_m$.	$\mathcal{B}^\top = \{x_i\}$	$\mathcal{M} \leftarrow \mathcal{M} \cup \{x_i\}$[2]
R_i^m	Reads the value x_i in the memory, and puts it into the top buffer. This operation takes a time $\text{cost}(R_i^m) = r_m$.	$x_i \in \mathcal{M}$	$\mathcal{B}^\top \leftarrow \{x_i\}$
W_i^d	Writes the value x_i of the top buffer into the disk. This operation takes a time $\text{cost}(W_i^d) = w_d$.	$\mathcal{B}^\top = \{x_i\}$	$\mathcal{D} \leftarrow \mathcal{D} \cup \{x_i\}$
R_i^d	Reads the value x_i in the disk and puts it into the top buffer. This operation takes a time $\text{cost}(R_i^d) = r_d$.	$x_i \in \mathcal{D}$	$\mathcal{B}^\top \leftarrow \{x_i\}$

Table 1: Operations performed by a schedule.

Because details of this instance are unimportant for this paper and would complicate the reading process, we refer the interested reader to our previous work (Definition 3.13, [1]) for those details. The main result of [1] is the dynamic programs to compute the value of $\text{Opt}_1^{(d)}(l, c_m, r_d)$ and $\text{Opt}_\infty(l, c_m, w_d, r_d)$ (see Theorem 1).

Theorem 1 (Theorems 1 and 3, [1])**.** *Let $l \in \mathbb{N}$, $c_m \in \mathbb{N}$, $w_d \in \mathbb{R}$ and $r_d \in \mathbb{R}$:*

$$Opt_\infty(l, c_m, w_d, r_d) = \min \begin{cases} Opt_0(l, c_m) \\ w_d + \min_{1\le j\le l-1}\{ju_f + Opt_\infty(l-j, c_m, w_d, r_d) + r_d + Opt_1^{(d)}(j-1, c_m, r_d)\} \end{cases}$$

$$Opt_1^{(d)}(l, c_m, r_d) = \min \begin{cases} Opt_0(l, c_m) \\ \min_{1\le j\le l-1}\{ju_f + Opt_0(l-j, c_m) + r_d + Opt_1^{(d)}(j-1, c_m, r_d)\} \end{cases}$$

Based on these dynamic programs, we designed a polynomial algorithm 1D-Revolve that, given the values l, c_m and r_d returns 1D-Revolve (l, c_m, r_d), an optimal sequence for $\overline{\text{Prob}}_1^{(d)}(l, c_m, r_d)$. This algorithm uses the binomial checkpointing algorithm Revolve, designed by Griewank and Walther [4], that solves the problem with only memory and no disk (Problem 3: $\text{Prob}(l, c_m)$ below). We note $\text{Opt}_0(l, c_m)$ the execution time of Revolve. We also defined Shift, the routine that takes a sequence S and an index `ind` and returns S shifted by `ind` (meaning for all $i \le l$, $s \in \{m, d\}$, W_i^s are replaced by $W_{i+\text{ind}}^s$, R_i^s are replaced by $R_{i+\text{ind}}^s$, F_i by $F_{i+\text{ind}}$, and $\bar{F}_i$ by $\bar{F}_{i+\text{ind}}$). Note that sequence Shift $(S, \texttt{ind})$ has the same execution time as sequence S.

Problem 3 ($\textsc{Prob}(l, c_m)$)**.** We want to minimize the makespan of the AC problem with the following parameters:

		Initial state:
AC graph:	size l	
Steps:	u_f, u_b	
Memory:	c_m, $w_m = r_m = 0$	$\mathcal{M}_{\text{ini}} = \emptyset$
Disks:	$c_d = 0$	
Buffers:	$\mathcal{B}^\top$, $\mathcal{B}^\bot$	$\mathcal{B}^\top_{\text{ini}} = \{x_0\}$, $\mathcal{B}^\bot_{\text{ini}} = \{\bar{x}_{l+1}\}$

Algorithm 1 Disk-Revolve

procedure Disk-Revolve(l, c_m, w_d, r_d)
  $\mathcal{S} \leftarrow \emptyset$
  if $\text{Opt}_\infty(l, c_m, w_d, r_d) = \text{Opt}_0(l, c_m)$ **then**
    $\mathcal{S} \leftarrow \textsc{Revolve}(l, c_m)$
  else
    Let j such that

$$\text{Opt}_\infty(l, c_m, w_d, r_d) = w_d + ju_f + \text{Opt}_\infty(l-j, c_m, w_d, r_d) + r_d + \text{Opt}_1^{(d)}(j-1, c_m, r_d)$$

    $\mathcal{S} \leftarrow W_0^d \cdot F_{0\to(j-1)}$
    $\mathcal{S} \leftarrow \mathcal{S} \cdot \textsc{Shift}\,(\textsc{Disk-Revolve}(l-j, c_m, w_d, r_d), j)$
    $\mathcal{S} \leftarrow \mathcal{S} \cdot R^d_{\text{ind}} \cdot \text{1D-}\textsc{Revolve}\,(j-1, c_m, r_d)$
  end if
  return $\mathcal{S}$
end procedure

Finally, we designed algorithm Disk-Revolve (Algorithm 1) that, given an adjoint computation graph of size $l \in \mathbb{N}$, $c_m \in \mathbb{N}$ memory slots, a cost $w_d \geq 0$ to write to disk and a cost $r_d \geq 0$ to read from disk, returns $\textsc{Disk-Revolve}(l, c_m, w_d, r_d)$ an optimal schedule for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. The time complexity of this algorithm is $O(l^2)$.

3 Dominant Sequences and Dominant Optimal Sequences

In this paper, we propose a different algorithm to compute solutions to $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. We introduce a family of sequences that we call *Dominant Sequences* (DS). A Dominant Sequence is a sequence that can be returned by algorithm AlgoDom (Algorithm 2). We call these sequences dominant, because we show in Section 5.3 that for every values l, c_m, r_d and w_d, there is a Dominant Sequence that is optimal for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. Such a sequence is called a Dominant Optimal Sequence (DOS).

To begin, we show that the sequence obtained using AlgoDom indeed returns a solution to $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$.

Algorithm 2 AlgoDom

1: **procedure** AlgoDom($m_1, \cdots, m_{n_l}$; res)
2: $\mathcal{S} \leftarrow \emptyset$
3: ind $\leftarrow 0$
4: **for** $i = 1$ to n_l **do**
5: $\mathcal{S} \leftarrow \mathcal{S} \cdot W^d_{\text{ind}} \cdot F_{\text{ind}\to(\text{ind}+m_i-1)}$
6: ind $\leftarrow$ ind $+ m_i$
7: **end for**
8: $\mathcal{S} \leftarrow \mathcal{S} \cdot$ Shift (Revolve(res $- 1, c_m$), ind)
9: **for** $i = n_l$ downto 1 **do**
10: $\mathcal{S} \leftarrow \mathcal{S} \cdot R^d_{\text{ind}} \cdot$ Shift (1D-Revolve $(m_i - 1, c_m, r_d)$, ind)
11: ind $\leftarrow$ ind $- m_i$
12: **end for**
13: **return** $\mathcal{S}$
14: **end procedure**

Definition 4 (Valid schedule)**.** A valid schedule is a sequence of operations from Table 1 such that all operations of the schedule are valid (they respect the input constraints).

Lemma 1. *Let* $(m_1, \cdots, m_{n_l}, res) \in \mathbb{N}^{n_l+1}$ *such that* $l = \sum_{i=1}^{n_l} m_i + res$*, then* AlgoDom $(m_1, \cdots, m_{n_l}; res)$ *is a valid schedule for* $\text{Prob}_\infty(l, c_m, w_d, r_d)$.

Proof. In order to do this we need to show that each operation done is authorized.

- The "for loop" on line 4 is correct as each F-operation is consecutive (so the needed information is in the top-buffer), and the $W^d_{\_}$ operations immediately follow the corresponding F-operation that puts the correct input in the top buffer.
- Applying Shift $\circ$ Revolve on line 8 is correct as at the beginning of this execution, (i) $x_{\text{ind}} = x_{l-\text{res}}$ is in the top buffer, (ii) $\bar{x}_{l+1}$ is in the bottom buffer and ind $= l -$ res, and Revolve(res, c_m) takes (i) x_0 in the top buffer and (ii) $\bar{x}_{\text{res}+1}$ in the bottom buffer. This sequence of operations returns $\bar{x}_{l-\text{res}}$ in the bottom buffer.
- The "for loop" on line 9 is correct. We show by induction $H(i)$: `at the beginning of the` i^{th} `iteration,` $\bar{x}_{\sum_{j=1}^{i+1} m_j}$ `is in the bottom buffer` and simultaneously we show $H'(i)$: `iteration` i `is correct`.

  We have already shown that $H(n_l)$ was true. Assume it is true until the i^{th} iteration. Then at the beginning of the i^{th} iteration, ind $= \sum_{j=1}^{i} m_j$ and x_{ind} was indeed written on disk during the i^{th} iteration of the "for loop" on line 4. 1D-Revolve is possible because $\text{Opt}_1^{(d)}(m_i - 1, c_m, r_d)$ takes as input x_0 in the top buffer and stored on disk, and $\bar{x}_{m_i}$ in the top buffer. Hence showing $H'(i)$. Furthermore, the result is then: $\bar{x}_{\text{ind}} = \bar{x}_{\sum_{j=1}^{i} m_j}$ in the bottom buffer, which shows $H(i-1)$.

The result of the last iteration of the "for loop" on line 9 is then exactly $\bar{x}_0$, which shows the correctness of AlgoDom. □

We now formally define notations that are used throughout this paper.

Definition 5. Let $l \geq 0$, $n_l \geq 0$, and $m_1, \cdots, m_{n_l}$, res such that $l = \sum_{i=1}^{n_l} m_i +$ res, then

- We shorten the sequence of operation returned by $\text{AlgoDom}(m_1, \cdots, m_{n_l}; \text{res})$ by $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ ($\mathcal{S} = (; l)$ if $n_l = 0$). $\mathcal{S}$ is called a *Dominant Sequence* (DS) for $\text{Prob}_\infty(l, c_m, w_d, r_d)$.
- $\mathcal{E}xec(\mathcal{S})$ is the *cost* (or the *execution time*) of the sequence $\mathcal{S}$, that is to say the sum of all the operations cost in $\mathcal{S}$.
- $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ is a *Dominant Optimal Sequence* (DOS) for $\text{Prob}_\infty(l, c_m, w_d, r_d)$ if the sequence $\mathcal{S}$ is an optimal solution to $\text{Prob}_\infty(l, c_m, w_d, r_d)$, that is to say $\mathcal{E}xec(\mathcal{S}) = \text{Opt}_\infty(l, c_m, w_d, r_d)$
- Let $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ a DS, the $m_1, \cdots, m_{n_l}$ are called the *periods* of $\mathcal{S}$, res is called the *residual* and n_l is the *number of periods* of $\mathcal{S}$.

4 Main contributions

In this work, we consider a platform with c_m memory slots, a writing cost to disk w_d and a reading cost from disk r_d. Throughout the rest of the paper, we note all these platform parameters $X = (c_m, r_d, w_d)$.

The main contribution of this paper is the following theorem:

Theorem 2 (Weak periodicity)**.** *There exists l_X, i_X and m_X such that: for all $l \geq l_X$, there exists $n_l \geq i_X$, and $(m_1, \cdots, m_{n_l}, res) \in \mathbb{N}^{n_l+1}$ (with $l = \sum_{i=1}^{n_l} m_i + res$) such that $\text{AlgoDom}(m_1, \cdots, m_{n_l}; res)$ is optimal for $\text{Prob}_\infty(l, c_m, w_d, r_d)$ and*

$$\forall i < n_l - i_X, m_i = m_X \tag{3}$$

$$\sum_{i=n_l-i_X}^{n_l} m_i + res < l_X \tag{4}$$

This result says that for every problem size, there exists a Dominant Optimal Sequence such that all the periods m_i's are equal to m_X, except for a bounded number of them.

This bound, i_X, only depends on the parameters X of the architecture. Stronger results would be to give values for i_X and for m_X. We derive the value for m_X experimentally in Section 7.

A consequence of this result is:

Corollary 1 (Online algorithm)**.** *There exists an asymptotically optimal algorithm for the online version (that is when the size of the graph is not known before-hand) of Problem 1.*

5 Showing the periodicity

In this section we prove the main contributions of the paper. In order to do so, we first start by giving structural arguments on the two core algorithms of the optimal solution: REVOLVE and 1D-REVOLVE in Section 5.1. Then we introduce some properties on the execution time of Dominant Sequences (DS), before showing that for all size l of Adjoint Computation graph, there exists a Dominant Sequence that is optimal (Theorem 6).

In order to prove Theorem 2:

1. We first construct a set of admissible periods $\mathcal{M}_X^{(\text{Adm})}$ in Section 5.4.
2. Then we show that the admissible periods have a maximum size in Section 5.5, hence showing that the set $\mathcal{M}_X^{(\text{Adm})}$ is finite.
3. We then construct $\mathcal{M}_X \subset \mathcal{M}_X^{(\text{Adm})}$ such that for all AC graph, there exists a DOS with periods solely in $\mathcal{M}_X$ in Section 5.6.
4. Finally, we show the result by contructing m_X, a period that is *dominant* over all other periods of $\mathcal{M}_X$.

5.1 Properties of a period

In this Section we give structural arguments on 1D-REVOLVE. In particular we are interested by the number of forward steps done by 1D-REVOLVE before the first memory write.

By definition 1D-REVOLVE can either behave as REVOLVE (and not use the disk checkpoint at hand), or do a certain number of disk reads. We discuss properties of the execution times of REVOLVE ($\text{Opt}_0(l, c_m)$) in Section 5.1.1 and of 1D-REVOLVE in general ($\text{Opt}_1^{(d)}(l, c_m, r_d)$) in Section 5.1.2.

5.1.1 Properties of $\text{Opt}_0(l, c_m)$

For $\text{Opt}_0(l, c_m)$, along with new results, we report existing results from Griewank and Walther [4] that we are using in this work. Note that most results from Griewank and Walther are adapted to the context here: in their work they only considered the number of re-execution of forward steps, while here we count the total execution time. In general this simply adds a cost of $lu_f + (l+1)u_b$ to the execution time.

Definition 6 (β)**.** We call β the function

$$\beta : x, y \mapsto \binom{x+y}{x} \tag{5}$$

Note that β is a critical parameter for all results from Griewank and Walther [4].

Lemma 2 (Griewank and Walther [4])**.** *Let $c_m \in \mathbb{N}$, then $x \in \mathbb{N} \mapsto Opt_0(x, c_m)$ is convex.*

Theorem 3 (Griewank and Walther [4] (Proposition 1, Equation 3))**.** *Let $l \in \mathbb{N}$ and $c_m \in \mathbb{N}$. The explicit form for $Opt_0(l, c_m)$ is:*

$$Opt_0(l, c_m) = l \cdot (t+1)u_f - \beta(c_m + 1, t-1)u_f + (l+1)u_b,$$

where t is the unique integer satisfying $\beta(c_m, t-1) \leq l < \beta(c_m, t)$.

Theorem 4 (Griewank and Walther [4] (Proposition 1, Equation 4))**.** *Let $l \in \mathbb{N}$ and $c_m \in \mathbb{N}$. Let t be the unique integer satisfying $\beta(c_m, t-1) \leq l < \beta(c_m, t)$. Denote j such that*

$$Opt_0(l, c_m) = ju_f + Opt_0(l - j, c_m - 1) + Opt_0(j - 1, c_m),$$

then

$$\beta(c_m, t-2) \leq j \leq \beta(c_m, t-1). \tag{6}$$

Lemma 3. *Let $l \in \mathbb{N}$, $c_m \in \mathbb{N}$ and t the unique integer satisfying $\beta(c_m, t-1) < l \leq \beta(c_m, t)$. Then:*

$$\begin{aligned} Opt_0(l+1, c_m) - Opt_0(l, c_m) &= (t+1) \cdot u_f + u_b \qquad (\text{if } l < \beta(c_m, t)) \\ Opt_0(l+1, c_m) - Opt_0(l, c_m) &= (t+2) \cdot u_f + u_b \qquad (\text{if } l = \beta(c_m, t)) \end{aligned}$$

Proof. Let $l \in \mathbb{N}$, $c_m \in \mathbb{N}$ and t the unique integer satisfying $\beta(c_m, t-1) < l \leq \beta(c_m, t)$.

- If $l < \beta(c_m, t)$, then t is also the unique integer satisfying $\beta(c_m, t-1) < l+1 \leq \beta(c_m, t)$. The explicit form for $\mathrm{Opt}_0(l+1, c_m)$ (see Theorem 3) is:

  $$\mathrm{Opt}_0(l+1, c_m) = (l+1) \cdot (t+1) \cdot u_f - \beta(c_m + 1, t-1) \cdot u_f + (l+2) \cdot u_b$$

  Thus:

  $$\mathrm{Opt}_0(l+1, c_m) - \mathrm{Opt}_0(l, c_m) = (t+1) \cdot u_f + u_b$$

- If $l = \beta(c_m, t)$, then t satisfies $\beta(c_m, t) < l+1 \leq \beta(c_m, t+1)$. The explicit form for $\mathrm{Opt}_0(l+1, c_m)$ (see Theorem 3) is:

  $$\mathrm{Opt}_0(l+1, c_m) = (l+1) \cdot (t+2) \cdot u_f - \beta(c_m + 1, t) \cdot u_f + (l+2) \cdot u_b$$

  Thus:

  $$\begin{aligned} \mathrm{Opt}_0(l+1, c_m) - \mathrm{Opt}_0(l, c_m) &= (l + t + 2 + \beta(c_m + 1, t-1) - \beta(c_m + 1, t)) \cdot u_f + u_b \\ &= (l + t + 2 - \beta(c_m, t)) \cdot u_f + u_b \\ &= (t+2) \cdot u_f + u_b \end{aligned}$$

□

Corollary 2. *Let $c_m \in \mathbb{N}$ and $t \in \mathbb{N}$.*
Then for all $l \geq \beta(c_m, t)$:

$$Opt_0(l+1, c_m) - Opt_0(l, c_m) \geq (t+2) \cdot u_f + u_b$$

Also, for all $l < \beta(c_m, t)$:

$$Opt_0(l+1, c_m) - Opt_0(l, c_m) \leq (t+1) \cdot u_f + u_b$$

Proof. Obviously t is the unique integer satisfying $\beta(c_m, t-1) < \beta(c_m, t) \leq \beta(c_m, t)$. Thus for all $t \in \mathbb{N}$:

$$\mathrm{Opt}_0(\beta(c_m, t)+1, c_m) - \mathrm{Opt}_0(\beta(c_m, t), c_m) = (t+2) \cdot u_f + u_b$$

- Let $l \geq \beta(c_m, t)$. By convexity of function $x \mapsto \mathrm{Opt}_0(x, c_m)$ (Lemma 2):

$$\begin{aligned}\mathrm{Opt}_0(l+1, c_m) - \mathrm{Opt}_0(l, c_m) &\geq \mathrm{Opt}_0(\beta(c_m, t)+1, c_m) - \mathrm{Opt}_0(\beta(c_m, t), c_m) \\ &\geq (t+2) \cdot u_f + u_b\end{aligned}$$

- Let $l < \beta(c_m, t)$. If $l > \beta(c_m, t-1)$ then t is the unique integer satisfying $\beta(c_m, t-1) < l \leq \beta(c_m, t)$ and:

$$\mathrm{Opt}_0(l+1, c_m) - \mathrm{Opt}_0(l, c_m) = (t+1) \cdot u_f + u_b$$

  If $l \leq \beta(c_m, t-1)$, By convexity of function $x \mapsto \mathrm{Opt}_0(x, c_m)$:

$$\begin{aligned}\mathrm{Opt}_0(l+1, c_m) - \mathrm{Opt}_0(l, c_m) &\geq \mathrm{Opt}_0(\beta(c_m, t-1)+1, c_m) - \mathrm{Opt}_0(\beta(c_m, t-1), c_m) \\ &\geq (t+1) \cdot u_f + u_b\end{aligned}$$

□

5.1.2 Properties of $\mathbf{Opt}_1^{(d)}(l, c_m, r_d)$

Contrarily to $\mathrm{Opt}_0(l, c_m)$, $\mathrm{Opt}_1^{(d)}(l, c_m, r_d)$ is not convex. This is mainly due to the fact that depending on the length of a period the number of disk reads can be different. However, we can extract some convexity from generalized functions where the number of disk reads is constant. This is the idea of function g_n presented in Definition 7 which is the execution time of a period if we were to do exactly n disk reads.

Definition 7. In the following, for a fixed $c_m \in \mathbb{N}$, $w_d \in \mathbb{R}$ and $r_d \in \mathbb{R}$, let us denote:

$$g_0(l) = \mathrm{Opt}_0(l, c_m) \tag{7}$$

$$\forall n > 0, \quad g_n(l) = \min_{0<j<l} \{g_{n-1}(j-1) + j u_f + \mathrm{Opt}_0(l-j, c_m) + r_d\} \tag{8}$$

We can show formally that $\mathrm{Opt}_1^{(d)}(l, c_m, r_d)$ is the optimum execution time among the solutions that do exactly $n \geq 0$ disk reads.

Lemma 4. *For all l,*

$$Opt_1^{(d)}(l, c_m, r_d) = \min_{n\geq 0} g_n(l)$$

Proof. We show the result by induction on l. For $l = 1$, then $\mathrm{Opt}_1^{(d)}(1, c_m, r_d) = \mathrm{Opt}_0(1, c_m) = g_0(1)$ and for all $n > 0$, $g_n(1) = +\infty$. Assume that for all $j < l$, $g(j) = \min_{n\geq 0} g_n(j)$ and let us prove the result for l. By definition of $\mathrm{Opt}_1^{(d)}(l, c_m, r_d)$ (Theorem 1),

$$\begin{aligned}
\mathrm{Opt}_1^{(d)}(l, c_m, r_d) &= \min_{0<j<l} \left\{ \begin{array}{l} \mathrm{Opt}_0(l, c_m) \\ ju_f + \mathrm{Opt}_0(l-j, c_m) + r_d + \mathrm{Opt}_1^{(d)}(j-1, c_m, r_d) \end{array} \right. \\
&= \min_{0<j<l} \left\{ \begin{array}{l} g_0(l) \\ ju_f + \mathrm{Opt}_0(l-j, c_m) + r_d + \min_{n\geq 0} g_n(j-1) \end{array} \right. \\
&= \min_{n\geq 0} \left\{ \begin{array}{l} g_0(l) \\ \min_{0<j<l}\{ju_f + \mathrm{Opt}_0(l-j, c_m) + r_d + g_n(j-1)\} \end{array} \right. \\
&= \min_{n\geq 0} \left(g_0(l), g_{n+1}(l)\right) \\
&= \min_{n\geq 0} g_n(l)
\end{aligned}$$

□

Lemma 5. *Let $c_m \in \mathbb{N}$ and $r_d \in \mathbb{R}$. For all $l \in \mathbb{N}$ and $n > 0$, the function*

$$h_{n,l} : j \mapsto ju_f + Opt_0(l-j, c_m) + r_d + g_{n-1}(j-1) \tag{9}$$

is convex.

Proof. We first show that g_n is convex. Clearly g_0 is convex (Lemma 2). We then show the result by induction on n. Assume for $n > 0$ that g_{n-1} is convex. By definition, g_n is the infimal convolution[3] of function $x \mapsto g_{n-1}(x)+(x+1)u_f+r_d$ and $l \mapsto \mathrm{Opt}_0(l, c_m)$ which are both convex. Hence for all n, g_n is convex [6]. So is $h_{n,l}$ as sum of convex functions. □

Finally, we can give an estimate of the number of forward steps done by 1D-Revolve when it does not behave as Revolve:

Lemma 6. *Given l, such that $Opt_1^{(d)}(l, c_m, r_d) < Opt_0(l, c_m)$. Consider $n^\star > 0$ and the largest $j^\star$ such that*

$$Opt_1^{(d)}(l, c_m, r_d) = h_{n^\star,l}(j^\star),$$

where $h_{n^\star,l}$ is defined by Equation (9). Then:

$$j^\star \geq \beta(c_m, t-1),$$

[3]The infimal convolution of two functions f and g is defined as $f\square g(l) = \min\{f(l-x) + g(x) | x \in \{0, \dots l\}\}$

where t is the unique integer satisfying $\beta(c_m, t-1) < \frac{l}{2} \leq \beta(c_m, t)$.

Proof. Since function $h_{n^\star,l}$ is convex (Lemma 5), its minimum is reached on a segment. Thus, we just have to prove that $h_{n^\star,l}(\beta(c_m,t-1)-1) \geq h_{n^\star,l}(\beta(c_m,t-1))$ to prove that the largest element of this segment (namely $j^\star$) is larger than $\beta(c_m, t-1)$. For simplicity let us note $y = \beta(c_m, t-1)$.

Assume first that $n^\star = 1$.

$$h_{1,l}(y-1) - h_{1,l}(y) = \\ -u_f + \mathrm{Opt}_0(l-y+1, c_m) - \mathrm{Opt}_0(l-y, c_m) + \mathrm{Opt}_0(y-2, c_m) - \mathrm{Opt}_0(y-1, c_m)$$

Because $y < \frac{l}{2}$, then $l - y > \frac{l}{2} > \beta(c_m, t-1)$. By Corollary 2:

$$\mathrm{Opt}_0(l-y+1, c_m) - \mathrm{Opt}_0(l-y, c_m) \geq (t+1)u_f + u_b.$$

Besides since $y = \beta(c_m, t-1)$, then $y - 2 < \beta(c_m, t-1)$. By Corollary 2:

$$\mathrm{Opt}_0(y-1, c_m) - \mathrm{Opt}_0(y-2, c_m) \leq t \cdot u_f + u_b.$$

Finally, we get:

$$\begin{aligned} h_{1,l}(y-1) - h_{1,l}(y) &\geq -u_f + (t+1)u_f + u_b - (t \cdot u_f + u_b) \\ &\geq 0, \end{aligned}$$

which proves that $j^\star \geq \beta(c_m, t-1)$.

Let us now assume that $n^\star \geq 2$.

$$h_{1,l}(y-1) - h_{1,l}(y) = \\ -u_f + \mathrm{Opt}_0(l-y+1, c_m) - \mathrm{Opt}_0(l-y, c_m) + g_{n-1}(y-2) - g_{n-1}(y-1)$$

Let $j \in \{1, ..., y-3\}$ such that:

$$g_{n-1}(y-2) = g_{n-2}(j-1) + ju_f + \mathrm{Opt}_0(y-2-j, c_m) + r_d$$

Clearly, since $j \in \{1, ..., y-2\}$:

$$g_{n-1}(y-1) \leq g_{n-2}(j-1) + ju_f + \mathrm{Opt}_0(y-1-j, c_m) + r_d$$

Thus:

$$h_{1,l}(y-1) - h_{1,l}(y) = \\ -u_f + \mathrm{Opt}_0(l-y+1, c_m) - \mathrm{Opt}_0(l-y, c_m) + \mathrm{Opt}_0(y-2-j, c_m) - \mathrm{Opt}_0(y-1-j, c_m)$$

Because $y < \frac{l}{2}$, then $l - y > \frac{l}{2} > \beta(c_m, t-1)$. By Corollary 2:

$$\mathrm{Opt}_0(l-y+1, c_m) - \mathrm{Opt}_0(l-y, c_m) \geq (t+1)u_f + u_b.$$

Besides since $y = \beta(c_m, t-1)$, then $y - 2 - j < \beta(c_m, t-1)$. By Corollary 2:

$$\mathrm{Opt}_0(y-1-j, c_m) - \mathrm{Opt}_0(y-2-j, c_m) \leq t \cdot u_f + u_b.$$

Finally, we get:

$$\begin{aligned} h_{1,l}(y-1) - h_{1,l}(y) &\geq -u_f + (t+1)u_f + u_b - (t \cdot u_f + u_b) \\ &\geq 0, \end{aligned}$$

which proves that $j^\star \geq \beta(c_m, t-1)$. □

Finally, we present another useful result: if 1D-Revolve behave as Revolve for a given l, then it will also do the same for all $l' < l$:

Lemma 7. *For all l, if $Opt_1^{(d)}(l, c_m, r_d) = Opt_0(l, c_m)$, then for all $l' < l$, $Opt_1^{(d)}(l', c_m, r_d) = Opt_0(l', c_m)$.*

Proof. We show the result by contradiction. Assume there exists $l' < l$ such that $\mathrm{Opt}_1^{(d)}(l', c_m, r_d) < \mathrm{Opt}_0(l', c_m)$. Then by Theorem 1, there exists $j < l'$ and

$$\mathrm{Opt}_1^{(d)}(l', c_m, r_d) = ju_f + \mathrm{Opt}_0(l'-j, c_m) + r_d + \mathrm{Opt}_1^{(d)}(j-1, c_m, r_d) < \mathrm{Opt}_0(l', c_m) \tag{10}$$

Because $\mathrm{Opt}_1^{(d)}(l, c_m, r_d) = \mathrm{Opt}_0(l, c_m)$, then by Theorem 1

$$\mathrm{Opt}_0(l, c_m) \leq ju_f + \mathrm{Opt}_0(l-j, c_m) + r_d + \mathrm{Opt}_1^{(d)}(j-1, c_m, r_d). \tag{11}$$

Finally, combining Eq. (10) and (11) we get:

$$\mathrm{Opt}_0(l, c_m) - \mathrm{Opt}_0(l', c_m) < \mathrm{Opt}_0(l-j, c_m) - \mathrm{Opt}_0(l'-j, c_m)$$

which is absurd by convexity of $l \mapsto \mathrm{Opt}_0(l, c_m)$ (Lemma 2). □

5.2 Execution time of a DS

In this section, we provide several basic results on the execution times of the Dominant Sequences that will be useful in the rest of the paper. We remind that given $\mathcal{S} = (m_1, \cdots, m_{n_l}; \mathrm{res})$ a DS, we denote by $\mathcal{E}xec(\mathcal{S})$ or $\mathcal{E}xec(m_1, \cdots, m_{n_l}; \mathrm{res})$ its execution time.

Theorem 5. *Let $\mathcal{S} = (m_1, \cdots, m_{n_l}; res)$ a DS and $l = \sum_{i=1}^{n_l} m_i + res$. Its execution time is:*

$$\mathcal{E}xec\,(\mathcal{S}) = (l - res)u_f + Opt_0(res, c_m) + \sum_{i=1}^{n_l} \Big(w_d + r_d + Opt_1^{(d)}(m_i - 1, c_m, r_d) \Big) \tag{12}$$

Proof. By definition of Algorithm 2, the execution time of $\mathcal{S}$ is:

$$\begin{aligned}\mathcal{E}xec(\mathcal{S}) =& \sum_{i=1}^{n_l}(w_d + m_i u_f) + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\Big(r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\\ =&(l - \mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\Big(r_d + w_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\end{aligned}$$

□

Corollary 3. *Given $n_l \geq 1$ and $\mathcal{S} = (m_1, \cdots, m_{n_l}; res)$ a DS. Then:*

$$\mathcal{E}xec\,(\mathcal{S}) = m_1 u_f + w_d + r_d + Opt_1^{(d)}(m_1 - 1, c_m, r_d) + \mathcal{E}xec\,(m_2, \cdots, m_{n_l}; res) \quad (13)$$

Proof. Note first that by definition, $(m_2, \cdots, m_{n_l}; \mathrm{res})$ is a DS for $\textsc{Prob}_\infty(l - m_1, c_m, w_d, r_d)$. Then we have:

$$\begin{aligned}\mathcal{E}xec\,(\mathcal{S}) =&(l - \mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\\ =&(l - m_1 - \mathrm{res})u_f + m_1 u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\\ =&m_1 u_f + \Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_1 - 1, c_m, r_d)\Big) + (l - m_1 - \mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m)\\ &+ \sum_{i=2}^{n_l}\Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\\ =&m_1 u_f + w_d + r_d + \mathrm{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) + \mathcal{E}xec\,(m_2, \cdots, m_{n_l}; \mathrm{res})\end{aligned}$$

□

Corollary 4. *Let $l \in \mathbb{N}$, $\mathcal{S} = (m_1, \cdots, m_{n_l}; res)$ a DS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$, and $\mathcal{S}_{+1} = (m_1, \cdots, m_{n_l}; res + 1)$, then*

$$\mathcal{E}xec(\mathcal{S}_{+1}) - \mathcal{E}xec(\mathcal{S}) = Opt_0(res + 1, c_m) - Opt_0(res, c_m) \quad (14)$$

Proof. This is a direct consequence of Equation (12):

$$\begin{aligned}\mathcal{E}xec\,(\mathcal{S}) &= (l - \mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\\ \mathcal{E}xec\,(\mathcal{S}_{+1}) &= ((l + 1) - (\mathrm{res} + 1))u_f + \mathrm{Opt}_0(\mathrm{res} + 1, c_m) + \sum_{i=1}^{n_l}\Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\Big)\end{aligned}$$

□

5.3 Existence of a DOS

In this section, we prove that for every values l, c_m, r_d and w_d there is a Dominant Sequence returned by AlgoDom that is optimal for $\text{Prob}_\infty(l, c_m, w_d, r_d)$. These sequences are called Dominant Optimal Sequences (DOS).

Theorem 6. *Given an adjoint computation graph of size $l \in \mathbb{N}$, $c_m \in \mathbb{N}$ memory slots, a cost $w_d \geq 0$ to write to disk and a cost $r_d \geq 0$ to read from disk, then there exists $n_l \in \mathbb{N}$ and $(m_1, \cdots, m_{n_l}, res) \in \mathbb{N}^{n_l+1}$, such that:*

- $l = \sum_{i=1}^{n_l} m_i + res$;
- *Sequence* $\text{AlgoDom}(m_1, \cdots, m_{n_l}; res)$ *is optimal for* $\text{Prob}_\infty(l, c_m, w_d, r_d)$.

Proof. We now show the existence of $n_l \in \mathbb{N}$ and $(m_1, \cdots, m_{n_l}, \text{res}) \in \mathbb{N}^{n_l+1}$ such that $l = \sum_{i=1}^{n_l} m_i + \text{res}$ and $\mathcal{E}xec(m_1, \cdots, m_{n_l}; \text{res}) = \text{Opt}_\infty(l, c_m, w_d, r_d)$.

First let us remind the optimal execution time for $\text{Prob}_\infty(l, c_m, w_d, r_d)$ (Theorem 1):

$$\text{Opt}_\infty(l, c_m, w_d, r_d) = \min \begin{cases} \text{Opt}_0(l, c_m) \\ w_d + \min_{1 \leq j \leq l-1} \{ j u_f + \text{Opt}_\infty(l-j, c_m, w_d, r_d) + r_d + \text{Opt}_1^{(d)}(j-1, c_m, r_d) \} \end{cases}$$

$$\text{Opt}_1^{(d)}(l, c_m, r_d) = \min \begin{cases} \text{Opt}_0(l, c_m) \\ \min_{1 \leq j \leq l-1} \{ j u_f + \text{Opt}_0(l-j, c_m) + r_d + \text{Opt}_1^{(d)}(j-1, c_m, r_d) \} \end{cases}$$

We now show the main result by induction on l. For $l = 1$, then $\text{Opt}_\infty(l, c_m, w_d, r_d) = \text{Opt}_0(l, c_m)$ and $n_l = 0$, and $(; l)$ is optimal for $\text{Prob}_\infty(l, c_m, w_d, r_d)$. Assume the result is true for all $l' < l$. Let us show the result for l.

- If $\text{Opt}_\infty(l, c_m, w_d, r_d) = \text{Opt}_0(l, c_m)$ then we choose $n_l = 0$ and $(; l)$ is optimal for $\text{Prob}_\infty(l, c_m, w_d, r_d)$.
- Otherwise, then there exists m_1, such that

  $$\text{Opt}_\infty(l, c_m, w_d, r_d) = w_d + m_1 u_f + \text{Opt}_\infty(l - m_1, c_m, w_d, r_d) + r_d + \text{Opt}_1^{(d)}(m_1 - 1, c_m, r_d).$$

  By induction hypothesis, there exists $(m_2, \cdots, m_{n_{l-m_1}+1}, \text{res})$ such that

  - $\sum_{i=2}^{n_{l-m_1}+1} m_i + \text{res} = l - m_1$
  - $(m_2, \cdots, m_{n_{l-m_1}+1}; \text{res})$ is optimal for $\text{Prob}_\infty(l - m_1, c_m, w_d, r_d)$.

  In particular, we have:

  $$\begin{aligned} \text{Opt}_\infty(l, c_m, w_d, r_d) &= w_d + m_1 u_f + \text{Opt}_\infty(l - m_1, c_m, w_d, r_d) + r_d + \text{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) \\ &= w_d + m_1 u_f + \mathcal{E}xec\left(m_2, \cdots, m_{n_{l-m_1}+1}; \text{res}\right) + r_d + \text{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) \\ &= \mathcal{E}xec(m_1, \cdots, m_{n_l}; \text{res}) \end{aligned}$$

The last equality is a consequence of Corollary 3. Finally, this shows that n_l, $(m_1, m_2, \cdots, m_{n_{l-m_1}+1}, \text{res})$ are such that $(m_1, \cdots, m_{n_l}; \text{res})$ is optimal for $\text{PROB}_\infty(l, c_m, w_d, r_d)$, hence showing the result.

□

Corollary 5. *Let $n_l \geq 1$ and $\mathcal{S} = (m_1, \cdots, m_{n_l}; res)$ a DOS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$ with $l = \sum_{i=1}^{n_l} m_i + res$. Then:*

1. *For all permutation $\sigma \in S_{n_l}$, the dominant sequence $\mathcal{S}_\sigma = (m_{\sigma(1)}, \cdots, m_{\sigma(n_l)}; res)$ is a DOS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$.*
2. *The dominant sequence $\mathcal{S}_{-m_1} = (m_2, \cdots, m_{n_l}; res)$ is a DOS for $\text{PROB}_\infty(l - m_1, c_m, w_d, r_d)$.*
3. *Given n_{l-m_1} and $\mathcal{S}' = (m'_1, \cdots, m'_{n_{l-m_1}}; res')$ a DOS for $\text{PROB}_\infty(l - m_1, c_m, w_d, r_d)$, then $\tilde{\mathcal{S}} = (m_1, m'_1, \cdots, m'_{n_{l-m_1}}; res')$ is a DOS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$.*

Proof. Based on Equation (12), $\mathcal{S}_\sigma$ and $\mathcal{S}$ have the same execution time. Thus, if $\mathcal{S}$ is an optimal solution to $\text{PROB}_\infty(l, c_m, w_d, r_d)$, so is $\mathcal{S}_\sigma$.

We prove the two next points together. First, by definition $\mathcal{S}_{-m_1}$ is a DS for $\text{PROB}_\infty(l - m_1, c_m, w_d, r_d)$. Because $\mathcal{S}'$ is a DOS for $\text{PROB}_\infty(l - m_1, c_m, w_d, r_d)$, then by optimality,

$$\mathcal{E}xec(\mathcal{S}') - \mathcal{E}xec(\mathcal{S}_{-m_1}) \leq 0.$$

Furthermore, by definition, $\tilde{\mathcal{S}}$ is a DS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$, hence

$$\mathcal{E}xec(\mathcal{S}) - \mathcal{E}xec(\tilde{\mathcal{S}}) \leq 0.$$

According to Corollary 3,

$$\mathcal{E}xec(\tilde{\mathcal{S}}) = m_1 u_f + w_d + r_d + \text{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) + \mathcal{E}xec(\mathcal{S}')$$
$$\mathcal{E}xec(\mathcal{S}) = m_1 u_f + w_d + r_d + \text{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) + \mathcal{E}xec(\mathcal{S}_{-m_1})$$

Hence, $0 \leq \mathcal{E}xec(\tilde{\mathcal{S}}) - \mathcal{E}xec(\mathcal{S}) = \mathcal{E}xec(\mathcal{S}') - \mathcal{E}xec(\mathcal{S}_{-m_1}) \leq 0$, and finally we obtain $\mathcal{E}xec(\tilde{\mathcal{S}}) = \mathcal{E}xec(\mathcal{S})$ and $\mathcal{E}xec(\mathcal{S}') = \mathcal{E}xec(\mathcal{S}_{-m_1})$. Hence, $\tilde{\mathcal{S}}$ is a DOS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$, and $\mathcal{S}_{-m_1}$ is a DOS for $\text{PROB}_\infty(l - m_1, c_m, w_d, r_d)$. □

5.4 Admissible periods $\mathcal{M}_X^{(\text{Adm})}$

Definition 8 (Admissible periods, $\mathcal{M}_X^{(\text{Adm})}$)**.** We call the set of admissible periods $\mathcal{M}_X^{(\text{Adm})}$:

$$\mathcal{M}_X^{(\text{Adm})} = \{m \in \mathbb{N} | \text{ there exists } l \text{ and } \mathcal{S} \text{ a DOS for } \text{PROB}_\infty(l, c_m, w_d, r_d) \text{ s.t. } m \text{ is a period of } \mathcal{S}\}$$

In this Section, we want to show properties on the set of admissible periods, in particular our goal is to show that it is bounded independently of the size of the adjoint computation.

Definition 9 (Decomposition)**.** We say that m can be decomposed into $[[m_1 : m_2]]$ if, $m = m_1 + m_2$ and

$$\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) > \mathrm{Opt}_1^{(d)}(m_1-1, c_m, r_d) + w_d + r_d + \mathrm{Opt}_1^{(d)}(m_2-1, c_m, r_d) \tag{15}$$

Intuitively, this means that there will be no DOS with a period of size m, two periods of size m_1, m_2 will be preferred. Let us write formally this result.

Theorem 7 (Decomposition)**.** *If m can be decomposed into $[[m_1 : m_2]]$, then there are no DOS with a period of size m (i.e., $m \notin \mathcal{M}_X^{(Adm)}$).*

Proof. We show the result by contradiction. Let m that can be decomposed into $[[m_1 : m_2]]$. Assume there exists l that admits a DOS with a period equal to m. Without loss of generality, let us denote $\mathcal{S} = (m, m_3, \cdots, m_{n_l}; \mathrm{res})$ this DOS.

Clearly, $\tilde{\mathcal{S}} = (m_1, \cdots, m_{n_l}; \mathrm{res})$ is a DS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. Let us show that $\mathcal{E}xec(\mathcal{S}) > \mathcal{E}xec(\tilde{\mathcal{S}})$ which would contradict the optimality of $\mathcal{S}$.

$$\begin{aligned}
\mathcal{E}xec\left(\tilde{\mathcal{S}}\right) =& (l-\mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=1}^{n_l}\left(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\right) \\
<& (l-\mathrm{res})u_f + \mathrm{Opt}_0(\mathrm{res}, c_m) + \sum_{i=3}^{n_l}\left(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d)\right) \\
& + \left(w_d + r_d + \mathrm{Opt}_1^{(d)}(m-1, c_m, r_d)\right) \\
<& \mathcal{E}xec\left(\mathcal{S}\right)
\end{aligned}$$

Hence the result. □

This decomposition allows us to caracterize periods that are not admissible.

Corollary 6. *Given $m \in \mathcal{M}_X^{(Adm)}$. If*

$$Opt_1^{(d)}(m-1, c_m, r_d) = j u_f + Opt_0(m-1-j, c_m) + Opt_0(j-1, c_m), \quad \textit{then } j \leq \frac{w_d + r_d}{u_f}. \tag{16}$$

$$Opt_1^{(d)}(m-1, c_m, r_d) = j u_f + Opt_0(m-1-j, c_m) + r_d + Opt_1^{(d)}(j-1, c_m, r_d), \quad \textit{then } j \leq \frac{w_d}{u_f}. \tag{17}$$

Proof. We show the results by contradiction.

Let us consider Equation (16), i.e. $\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) = \mathrm{Opt}_0(m-1, c_m)$. Assume $j > \frac{w_d + r_d}{u_f}$. Because we have $\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) = \mathrm{Opt}_0(m-1, c_m)$, by Lemma 7,

$$\mathrm{Opt}_1^{(d)}(m-1-j, c_m, r_d) = \mathrm{Opt}_0(m-1-j, c_m) \qquad \text{and}$$
$$\mathrm{Opt}_1^{(d)}(j-1, c_m, r_d) = \mathrm{Opt}_0(j-1, c_m).$$

Consequently,

$$\mathrm{Opt}_1^{(d)}(m-1,c_m,r_d) > w_d+r_d+\mathrm{Opt}_1^{(d)}(m-1-j,c_m,r_d)+\mathrm{Opt}_1^{(d)}(j-1,c_m,r_d)$$

and m can be decomposed into $[[j : m-j]]$ and the result follows from Theorem 7.

Let us now consider Equation (17), i.e. $\mathrm{Opt}_1^{(d)}(m-1,c_m,r_d) < \mathrm{Opt}_0(m-1,c_m)$. Assume $j > \frac{w_d}{u_f}$, then similarly it suffices to see that m can be decomposed into $[[j : m-j]]$ (because $\mathrm{Opt}_0(m-1-j,c_m) \geq \mathrm{Opt}_1^{(d)}(m-1-j,c_m,r_d)$) and the result follows from Theorem 7.

□

Corollary 7. *Let $m \in \mathcal{M}_X^{(Adm)}$, then:*

$$Opt_1^{(d)}(m-1,c_m,r_d) = \min_{1\leq j\leq \min\left(\frac{w_d}{u_f},l-1\right)} \begin{cases} Opt_0(m-1,c_m) \\ ju_f + Opt_0(m-1-j,c_m) + r_d + Opt_1^{(d)}(j-1,c_m,r_d) \end{cases}$$

Proof. This is a direct corollary from Corollary 6 □

Note that while Corollary 7 is not directly involved in the proof of Theorem 2, it provides a faster execution of $\mathrm{Opt}_1^{(d)}(l,c_m,r_d)$.

5.5 Bounding the size of a period

The main result of this section is that, for a given platform (defined by its parameters c_m, w_d, and r_d) the set of admissible periods is actually bounded. We prove this result by showing that admissible periods admit an upper bound on their size.

5.5.1 Maximum period size

Lemma 8. *Let t_d^1 and t_d^2 be the only integers such that:*

$$\beta(c_m,t_d^1) \geq \frac{w_d+r_d}{u_f} > \beta(c_m,t_d^1-1),$$
$$\beta(c_m,t_d^2) \geq \frac{w_d}{u_f} > \beta(c_m,t_d^2-1),$$

then there are no admissible periods in $\mathcal{M}_X^{(Adm)}$ greater than

$$m_{\max} = \max(\beta(c_m,t_d^1+1), 2\beta(c_m,t_d^2)+1) \tag{18}$$

Proof. Let $m \geq m_{\max}$.

Assume first that $\mathrm{Opt}_1^{(d)}(m-1,c_m,r_d) = \mathrm{Opt}_0(m-1,c_m)$. Then $\mathrm{Opt}_1^{(d)}(m-1,c_m,r_d) = j_m u_f + \mathrm{Opt}_0(m-j_m-1,c_m) + \mathrm{Opt}_0(j-1,c_m)$, where $j_m \geq \beta(c_m,t_d^1) \geq \frac{w_d+r_d}{u_f}$ (Theorem 4). Then, by Corollary 6, $m \notin \mathcal{M}_X^{(\mathrm{Adm})}$.

Otherwise, assume $\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) < \mathrm{Opt}_0(m-1, c_m)$. According to Lemma 6 there exists $j_m \geq \beta(c_m, t_d^2) \geq \frac{w_d}{u_f}$ and n_m such that:

$$\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) = j_m u_f + \mathrm{Opt}_0(m - j_m - 1, c_m) + r_d + g_{n_m-1}(j_m - 1)$$

In particular, by Definition 7, we know that $g_{n_m-1}(j_m - 1) = \min_n g_n(j_m - 1)$. And by Lemma 4, $g_{n_m-1}(j_m - 1) = \mathrm{Opt}_1^{(d)}(j_m - 1, c_m, r_d)$. Thus:

$$\mathrm{Opt}_1^{(d)}(m-1, c_m, r_d) = j_m u_f + \mathrm{Opt}_0(m - j_m - 1, c_m) + r_d + \mathrm{Opt}_1^{(d)}(j_m - 1, c_m, r_d).$$

Then, by Corollary 6, $m \notin \mathcal{M}_X^{(\mathrm{Adm})}$. □

Theorem 8. *The number of admissible periods $|\mathcal{M}_X^{(Adm)}|$ is bounded.*

Proof. This theorem is a corollary of Lemma 8: there are at most $m_{\max}$ periods in $\mathcal{M}_X^{(\mathrm{Adm})}$ and $\mathcal{M}_X^{(\mathrm{Adm})} \cap \{m_{\max}+1, ...\} = \emptyset$. □

Corollary 8. *Let l, then:*

$$Opt_\infty(l, c_m, r_d, w_d) =$$
$$\min_{1 \leq j \leq m_{\max}} \begin{cases} Opt_0(m-1, c_m) \\ w_d + j u_f + Opt_\infty(m-1-j, c_m, r_d, w_d) + r_d + Opt_1^{(d)}(j-1, c_m, r_d) \end{cases}$$

Proof. This is a direct corollary from Lemma 8 □

Note that as for Corollary 7, Corollary 8 is not directly involved in the proof of Theorem 2, but provides a faster execution of $\mathrm{Opt}_\infty(l, c_m, w_d, r_d)$.

5.6 Construction of $\mathcal{M}_X$ and periodicity

In the previous section, we proved that, for a given platform (defined by its parameters c_m, w_d, and r_d) the set of admissible period that can be used by a DOS is finite. But this set can be quite large in practice. In this section, we define a sufficient subset $\mathcal{M}_X$ of $\mathcal{M}_X^{(\mathrm{Adm})}$ and highlight its element m_X such that for any adjoint computation graph of any size, there is an optimal solution with only periods of size m_X, except for a bounded number of them. We also provide, in this section, an algorithm to compute $\mathcal{M}_X$ and m_X.

5.6.1 Bounding the number of solutions with less than one period

Definition 10 ($\mathcal{N}_X(l)$,$l_X^{(1)}$, $l_X^{(2)}$)**.** Let $\mathcal{N}_X(l)$ the largest (by inclusion) set such that for all $n_l \in \mathcal{N}_X(l)$, there exists a DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ with n_l periods.

Let $l_X^{(1)}$ be the smallest integer such that for all $l \geq l_X^{(1)}$, $\mathcal{N}_X(l) \not\subset \{0\}$, that is, the smallest integer such that for all graph size greater than $l_X^{(1)}$, there exists a DOS with at least one period.

Let $l_X^{(2)}$ be the smallest integer such that for all $l \geq l_X^{(2)}$, $\mathcal{N}_X(l) \not\subset \{0,1\}$, that is, the smallest integer such that for all graph size greater than $l_X^{(2)}$, there exists a DOS with at least two periods.

Lemma 9. *Let $l \in \mathbb{N}$, $n_l \in \mathcal{N}_X(l)$, then*

$$n_l \geq 1 \implies 0 \notin \mathcal{N}_X(l+1).$$

Proof. Consider $l \in \mathbb{N}$, $n_l \in \mathcal{N}_X(l)$ such that $n_l \geq 1$. Let $\mathcal{S}_l = (m_1, \cdots, m_{n_l}; \text{res})$ be a DOS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$. $\mathcal{S}'_l = (; l)$ is a DS for $\text{PROB}_\infty(l, c_m, w_d, r_d)$. Thus by definition,

$$\mathcal{E}xec(\mathcal{S}_l) \leq \mathcal{E}xec(\mathcal{S}'_l).$$

Let $\mathcal{S}'_{l+1} = (; l+1)$ and $\mathcal{S}'_{l+1} = (m_1, \cdots, m_{n_{X_n}(l)}; \text{res}+1)$. They are both DS for $\text{PROB}_\infty(l+1, c_m, w_d, r_d)$.Then we have:

$$\begin{aligned}
&\mathcal{E}xec(\mathcal{S}_{l+1}) - \mathcal{E}xec(\mathcal{S}'_{l+1}) && \text{(19a)}\\
&\quad \leq \mathcal{E}xec(\mathcal{S}_{l+1}) - \mathcal{E}xec(\mathcal{S}'_{l+1}) + (\mathcal{E}xec(\mathcal{S}'_l) - \mathcal{E}xec(\mathcal{S}_l)) && \text{(19b)}\\
&\quad = (\mathcal{E}xec(\mathcal{S}_{l+1}) - \mathcal{E}xec(\mathcal{S}_l)) - \left(\mathcal{E}xec(\mathcal{S}'_{l+1}) - \mathcal{E}xec(\mathcal{S}'_l)\right)\\
&\quad = (\text{Opt}_0(\text{res}+1, c_m) - \text{Opt}_0(\text{res}, c_m)) - (\text{Opt}_0(l+1, c_m) - \text{Opt}_0(l, c_m)) && \text{(19c)}\\
&\quad < 0 && \text{(19d)}
\end{aligned}$$

Equation (19b) is because $\mathcal{E}xec(\mathcal{S}_l) \leq \mathcal{E}xec(\mathcal{S}'_l)$. Then we obtain Equation (19c) through Equation (14). Finally, Equation (19d) is because $l \mapsto \text{Opt}_0(l, c_m)$ is convex (Lemma 2).

Finally, $\mathcal{S}'$ cannot be a DOS because another DS has a better execution time. Hence, $0 \notin \mathcal{N}_X(l+1)$. □

Corollary 9. *For all $l \in \mathbb{N}$:*

$$\begin{aligned}
l < l_X^{(1)} &\implies \mathcal{N}_X(l) = \{0\}\\
l > l_X^{(1)} &\implies 0 \notin \mathcal{N}_X(l)
\end{aligned}$$

Proof. This is a corollary of Lemma 9. □

Lemma 10 (Existence of $l_X^{(1)}$)**.** *There exists $l > 0$, such that $0 \notin \mathcal{N}_X(l)$. Furthermore,*

$$l_X^{(1)} \leq \beta(c_m, t_d + 3) - \frac{\beta(c_m, t_d + 2)}{2}, \tag{20}$$

where t_d is the unique integer satisfying $\beta(c_m, t_d - 1) < \frac{2(w_d + r_d)}{u_f} \leq \beta(c_m, t_d)$.

Proof. Given $j \geq \beta(c_m, t_d + 1)$, let us show that

$$\mathrm{Opt}_0(j, c_m) > \min_{k<j} \Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(k-1, c_m, r_d) + w_d + r_d + \mathrm{Opt}_1^{(d)}(j-k, c_m, r_d) \Big) \tag{21}$$

According to Theorem 4, there exist $\tilde{j}$ such that: (i) $\beta(c_m, t_d) \leq \tilde{j}$, and (ii) $\mathrm{Opt}_0(j, c_m) = \tilde{j} u_f + \mathrm{Opt}_0(\tilde{j} - 1, c_m) + \mathrm{Opt}_0(j - \tilde{j}, c_m)$. Then,

$$\begin{aligned}
\mathrm{Opt}_0(j, c_m) &= \tilde{j} u_f + \mathrm{Opt}_0(\tilde{j} - 1, c_m) + \mathrm{Opt}_0(j - \tilde{j}, c_m - 1) \\
&\geq \beta(c_m, t_d) u_f + \mathrm{Opt}_0(\tilde{j} - 1, c_m) + \mathrm{Opt}_0(j - \tilde{j}, c_m) \\
&> 2(w_d + r_d) + \mathrm{Opt}_1^{(d)}(\tilde{j} - 1, c_m, r_d) + \mathrm{Opt}_1^{(d)}(j - \tilde{j}, c_m, r_d)
\end{aligned}$$

Showing the correctness of Equation (21).

Let us now consider $l \geq \beta(c_m, t_d + 3)$, then according to Theorem 4, there exist j such that: (i) $\beta(c_m, t_d + 2) \leq j$, and (ii) $\mathrm{Opt}_0(l, c_m) = j u_f + \mathrm{Opt}_0(j - 1, c_m) + \mathrm{Opt}_0(l - j, c_m - 1)$.

Let m_1, m_2 such that $m_1 + m_2 = j$ and

$$\begin{aligned}
&\mathrm{Opt}_1^{(d)}(m_1 - 1, c_m, r_d) + \mathrm{Opt}_1^{(d)}(m_2 - 1, c_m, r_d) \\
&\qquad = \min_{k<j-1} \mathrm{Opt}_1^{(d)}(k-1, c_m, r_d) + \mathrm{Opt}_1^{(d)}(j-1-k, c_m, r_d).
\end{aligned}$$

Clearly, $(m_1, m_2; l - j)$ is a DS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. Let us now show that $\mathcal{E}xec\,(m_1, m_2; l - j) < \mathcal{E}xec\,(; l)$ which shows the result.

$$\begin{aligned}
\mathcal{E}xec\,(m_1, m_2; l - j) &= j u_f + \mathrm{Opt}_0(l - j, c_m) + \sum_{i=1}^{2} \Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d) \Big) \\
&\leq j u_f + \mathrm{Opt}_0(l - j, c_m - 1) + \sum_{i=1}^{2} \Big(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_i - 1, c_m, r_d) \Big) \qquad (22a) \\
&< j u_f + \mathrm{Opt}_0(l - j, c_m - 1) + \mathrm{Opt}_0(j - 1, c_m) \qquad (22b) \\
&< \mathcal{E}xec\,(; l)
\end{aligned}$$

Equation (22a) is due to the fact that adding memory checkpoints can only improve the execution time, Equation (22b) is because of Equation (21).

Hence the result: for $l \geq \beta(c_m, t_d + 3)$, $0 \notin \mathcal{N}_X(l)$. Clearly we have: $l_X^{(1)} \leq \beta(c_m, t_d + 3)$. We can improve this result by saying that $(m_1; l - j)$ (resp. $(m_2; l - j)$) is a DOS for $\textsc{Prob}_\infty(l - m_2, c_m, w_d, r_d)$ (resp. $\textsc{Prob}_\infty(l - m_1, c_m, w_d, r_d)$) by Corollary 5. Furthermore, by Lemma 9, this implies that for all $l \geq \beta(c_m, t_d + 3) - \max(m_1 + m_2)$, $0 \notin \mathcal{N}_X(l)$. Hence, $l_X^{(1)} \leq \beta(c_m, t_d + 3) - \max(m_1 + m_2)$. Furthermore, we have seen that $m_1 + m_2 \geq \beta(c_m, t_d + 2)$ and in particular, $\max(m_1 + m_2) \geq \frac{\beta(c_m, t_d + 2)}{2}$.

Finally, we have the result,

$$l_X^{(1)} \leq \beta(c_m, t_d + 3) - \frac{\beta(c_m, t_d + 2)}{2}.$$

□

Based on Corollary 9 and Lemma 10, we can construct an algorithm to compute $l_X^{(1)}$. Corollary 9 says that Algorithm 3 returns the value of $l_X^{(1)}$, while Lemma 10 ensures that the algorithm terminates in less than $\beta(c_m, t_d + 3) - \frac{\beta(c_m, t_d+2)}{2}$ iterations of the "while" loop.

Algorithm 3 Algorithm to compute $l_X^{(1)}$

```
1: procedure AlgoL1X(c_m, w_d, r_d)
2:    l_X^(1) ← 1
3:    while for all j ≤ l_X^(1), Opt_0(l_X^(1), c_m) < w_d + j u_f + Opt_∞(l_X^(1) − 1 − j, c_m, w_d, r_d) +
      r_d + Opt_1^(d)(j − 1, c_m, r_d) do
4:       l_X^(1) ← l_X^(1) + 1
5:    end while
6:    return l_X^(1)
7: end procedure
```

Lemma 11 (Existence of $l_X^{(2)}$)**.**

$$l_X^{(2)} \leq l_X^{(1)} + m_{\max}, \tag{23}$$

where $m_{\max}$ *is defined by Equation* (18)*.*

Proof. Furthermore, let us show that for all $l \geq l_X^{(1)} + 2\beta(c_m, t_d)$, $1 \notin \mathcal{N}_X(l)$.

Let l such that $1 \in \mathcal{N}_X(l)$. Let $(m; l-m)$ be a DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. According to Lemma 8, $m \leq m_{\max}$. Furthermore, according to Corollary 5, $(; l-m)$ is a DOS for $\textsc{Prob}_\infty(l-m, c_m, w_d, r_d)$. Hence, according to Lemma 9, $l - m \leq l_X^{(1)}$. Finally, we have: $l \leq l_X^{(1)} + m_{\max}$. □

5.6.2 M-DOS and periodicity

In order to show the periodicity, we define a set $\mathcal{M}_X$ such that for all l, there exists a DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ such that all its periods are in $\mathcal{M}_X$. We call such DOS: M-DOS.

Definition 11 ($\mathcal{M}_X$, m_X)**.** For this section we use the following definitions:

- Let $\mathcal{L}_X$ the set of adjoint graphs that only admits DOS with one period:

$$\mathcal{L}_X = \left\{ l \middle| \mathcal{N}_X(l) = \{1\} \right\} \tag{24}$$

- We define the *relative cost of a period*[4] $\texttt{RelCost}_X$:

$$\texttt{RelCost}_X : m \mapsto \frac{w_d + r_d + \mathrm{Opt}_1^{(d)}(m-1, c_m, r_d)}{m}. \tag{25}$$

- For $l \in \mathcal{L}_X$, we define m_1^l to be the minimum element of

$$\{m | (m; l-m) \text{ is a DOS for } \textsc{Prob}_\infty(l, c_m, w_d, r_d)\}$$

  with regard to $\texttt{RelCost}_X$.
- We define a subset of the set of admissible periods $\mathcal{M}_X$:

$$\mathcal{M}_X = \left\{ m_1^l \middle| l \in \mathcal{L}_X \right\}. \tag{26}$$

- Denote m_X the largest element of $\mathcal{M}_X$ that is minimum with regard to $\texttt{RelCost}_X$.

Note that according to Corollary 9, $\mathcal{L}_X \subset \{l_X^{(1)}, \cdots, \infty\}$, and by Definition 10, $\mathcal{L}_X \subset \{1, \cdots, l_X^{(2)} - 1\}$. Hence

$$|\mathcal{M}_X| < l_X^{(2)} - l_X^{(1)}.$$

Corollary 9 provides an efficient way to check whether $l \in \mathcal{L}_X$:

Proposition 1. *$l \in \mathcal{L}_X$ if and only if*

$$Opt_\infty(l, c_m, w_d, r_d) < \min_{j \le l - l_X^{(1)}} \begin{cases} Opt_0(l, c_m) \\ w_d + j u_f + Opt_\infty(l-1-j, c_m, r_d, w_d) + r_d + Opt_1^{(d)}(j-1, c_m, r_d) \end{cases} \tag{27}$$

Proof. If $l \in \mathcal{L}_X$ then $\mathrm{Opt}_\infty(l, c_m, r_d, w_d) < \mathrm{Opt}_0(l, c_m)$ ($0 \notin \mathcal{N}_X(l)$). Furthermore, by Corollary 9, if a DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ admits a period m such that $l - m \ge l_X^{(1)}$, then $\mathcal{N}_X(l-m) \neq \{0\}$ and $l \notin \mathcal{L}_X$.

Finally, if $\mathrm{Opt}_0(l, c_m)$ satisfies Equation (27), then $\mathrm{Opt}_\infty(l, c_m, w_d, r_d) = w_d + j u_f + \mathrm{Opt}_\infty(l-1-j, c_m, r_d, w_d) + r_d + \mathrm{Opt}_1^{(d)}(j-1, c_m, r_d)$ only for j such that $l - j < l_X^{(1)}$, and by Corollary 9, $\mathcal{N}_X(l) = \{1\}$. □

Proposition 1 in turn provides an efficient way to compute m_X:

Definition 12 (M-DOS)**.** For all l, $\mathcal{S} = (m_1, \cdots, m_{n_l}; \mathrm{res})$ is a M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ if (i) $\mathcal{S}$ is a DOS and (ii), for all i, $m_i \in \mathcal{M}_X$.

Theorem 9 (Existence of M-DOS)**.** *Given $X = (c_m, r_d, w_d)$. For all $l \ge l_X^{(1)}$, there exists a M-DOS.*

[4] Intuitively, for a period of size m we pay w_d to be able to execute it later, and then $r_d + \mathrm{Opt}_1^{(d)}(m-1, c_m, w_d, r_d)$ to actually execute it.

Algorithm 4 Algorithm to compute m_X

```
1: procedure AlgomX(c_m, w_d, r_d)
2:    l_X^(1) = AlgoL1X(c_m, w_d, r_d)
3:    M_X ← {}
4:    for l = l_X^(1) ··· l_X^(1) + m_max do
5:       if l ∈ L_X then M_X ← M_X ∪ {m_1^l}
6:       end if
7:    end for
8:    m_X ← minimum of M_X with regard to RelCost_X
9:    return m_X
10: end procedure
```

Proof. We show the result by contradiction.

Let us call $l_0 \geq l_X^{(1)}$ the minimum length that does not admit a M-DOS, i.e., such that for all DOS for $\textsc{Prob}_\infty(l_0, c_m, w_d, r_d)$ there exists a period not in $\mathcal{M}_X$.

Amongst the DOS for $\textsc{Prob}_\infty(l_0, c_m, w_d, r_d)$ such that the number of periods is greater than 2 (by definition $l_0 \geq l_X^{(1)}$, and $l_0 \notin \mathcal{L}_X$ hence there exists at least one), we choose $\mathcal{S} = (m_1, \cdots, m_{n_{l_0}}; \text{res})$ a DOS that is minimum with regard to the number of periods not in $\mathcal{M}_X$.

By Corollary 5 (item 1) we can further assume w.l.o.g that $m_{n_{l_0}} \notin \mathcal{M}_X$. Iterating Corollary 5 (item 2), $(m_{n_{l_0}}; \text{res})$ is a DOS for $\text{Opt}_\infty(m_{n_{l_0}} + \text{res}, c_m, w_d, r_d)$.

Clearly, $m_{n_{l_0}} + \text{res} < l_0$ (there are more than two periods, hence $l_0 \geq m_1 + m_{n_{l_0}} + \text{res} > m_{n_{l_0}} + \text{res}$), and because $1 \in \mathcal{N}_X(m_{n_{l_0}} + \text{res})$, by Lemma 9, $m_{n_{l_0}} + \text{res} \geq l_X^{(1)}$. By minimality of l_0, there exists $\mathcal{S}'$ a DOS for $\text{Opt}_\infty(m_{n_{l_0}} + \text{res}, c_m, w_d, r_d)$ such that all periods of $\mathcal{S}'$ are in $\mathcal{M}_X$ and such that there is at least one period.

Finally, by Corollary 5 (item 3), we can replace $(m_{n_{l_0}}; \text{res})$ in $\mathcal{S}$ by $\mathcal{S}'$, then (i) it will still be an optimal algorithm for l_0, and (ii) it will have more than 2 periods.

Finally, we have a DOS for $\textsc{Prob}_\infty(l_0, c_m, w_d, r_d)$ with more than two periods and one less period than $\mathcal{S}$ not in $\mathcal{M}_X$ contradicting the existence of $\mathcal{S}$. □

Definition 13 ($M^{\mathcal{S}}_{-m_X}$)**.** Let $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ a M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$, we define

$$M^{\mathcal{S}}_{-m_X} = \sum_{i \in \{i | m_i \neq m_X\}} m_i.$$

Less formally, $M^{\mathcal{S}}_{-m_X}$ is the sum of the length of the periods of $\mathcal{S}$ that are not m_X.

Lemma 12. *For all $l \in \mathbb{N}$, there exists $\mathcal{S}$ a M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ such that, for all $m \in \mathcal{M}_X \setminus \{m_X\}$, there are less than m_X periods of size m.*

Proof. We show the result by contradiction. Let l such that for any given M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$, there exists $m \in \mathcal{M}_X \setminus \{m_X\}$ and there are not less than m_X periods of size m. Let $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ a M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ that is minimum with regards to $M^{\mathcal{S}}_{-m_X}$. Let $m \in \mathcal{M}_X \setminus \{m_X\}$ such that there are not less than m_X periods of size m in $\mathcal{S}$. Without loss of generality, we can assume that for $i \leq m_X$, $m_i = m$ (Corollary 5 (item 1), the m_X first periods have a size m).

According to Corollary 3 (and by induction on the m_X first elements),

$$\mathcal{E}xec\,(\mathcal{S}) = m_X \left(m u_f + w_d + r_d + \text{Opt}_1^{(d)}(m-1, c_m, r_d) \right) + \mathcal{E}xec\,(m_{m_X+1}, \ldots, m_{n_l}; \text{res})$$

Denote $\mathcal{S}' = (m'_1, \cdots, m'_m, m_{m_X+1}, \cdots, m_{n_l}; \text{res})$, where for $i \leq m$, $m'_i = m_X$. Then clearly, $\sum_{i=1}^{m} m_X = \sum_{i=1}^{m_X} m$, and then $\mathcal{S}'$ is a DS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$.

Furthermore,

$$\mathcal{E}xec\,(\mathcal{S}') = m \left(m_X u_f + w_d + r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right) + \mathcal{E}xec\,(m_{m_X+1}, \ldots, m_{n_l}; \text{res})$$

Then,

$$\mathcal{E}xec\,(\mathcal{S}) - \mathcal{E}xec\,(\mathcal{S}') = \\ m_X \left(w_d + r_d + \text{Opt}_1^{(d)}(m-1, c_m, r_d) \right) - m \left(w_d + r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right)$$

Hence, because m_X is minimal with regards to the function $\texttt{RelCost}_X$, $\texttt{RelCost}_X(m) \geq \texttt{RelCost}_X(m_X)$ and $\mathcal{E}xec\,(\mathcal{S}) - \mathcal{E}xec\,(\mathcal{S}') \geq 0$.

By optimality of $\mathcal{S}$, we have: $\mathcal{E}xec\,(\mathcal{S}) = \mathcal{E}xec\,(\mathcal{S}')$. However

$$M^{\mathcal{S}'}_{-m_X} = M^{\mathcal{S}}_{-m_X} - m_X \cdot m,$$

which contradicts the minimality of $\mathcal{S}$, proving the result. □

Proof of Theorem 2. Let:

$$i_X^{\text{ub}} = (m_X - 1) \cdot (|\mathcal{M}_X| - 1)$$
$$l_X^{\text{ub}} = l_X^{(1)} + i_X^{\text{ub}} m_{\max}.$$

Let $l \geq l_X^{\text{ub}}$, then according to Theorem 9 and Lemma 12, we can construct $\mathcal{S} = (m_1, \cdots, m_{n_l}; \text{res})$ a M-DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$ such that for any $m \in \mathcal{M}_X \setminus \{m_X\}$, at most $m_X - 1$ periods of size m. Hence, there are at most i_X^{ub} periods not equal to m_X, and $n_l - i_X^{\text{ub}}$ periods of size m_X.

According to Corollary 5 (item 1), we can assume that the first $n_l - i_X^{\text{ub}}$ periods are the periods of size m_X (hence showing Equation (3)).

Furthermore, according to Corollary 5 (item 2),

- $(m_{n_l - i_X^{\text{ub}}}, \cdots, m_{n_l}; \text{res})$ is a M-DOS with i_X^{ub} periods all smaller than $m_{\max}$;
- $(; \text{res})$ is a DOS hence $\text{res} < l_X^{(1)}$.

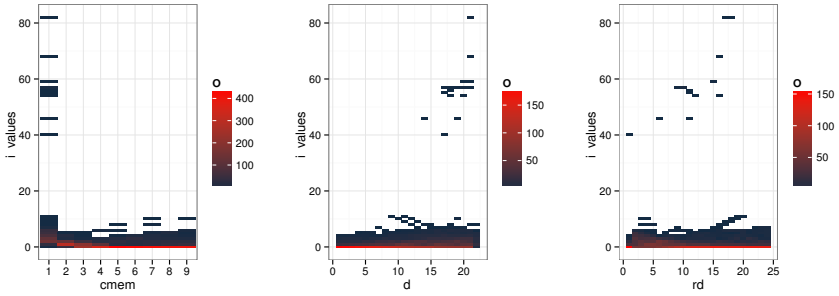


Figure 2: Heat maps of the values of i_X according to the three different platform parameters: c_m, w_d, r_d (white = 0 occurence). Most of the time, $i_X \leq 10$.

Finally, $\sum_{i=n_l-i_X^{\text{ub}}}^{n_l} m_i + \text{res} < l_X^{\text{ub}}$ (hence showing Equation (4)).

This shows the existence of i_X and l_X. □

In this proof, we have not given the smallest value possible for i_X and l_X, we have only given upper bounds to show their existence.

5.7 Experimental evaluation of i_X and m_X

Theorem 2 only gives a weak periodicity argument. In this Section we try to develop questions that can arise from Theorem 2.

5.7.1 Can we get a better bound for i_X?

However, one might expect that by improving the bounds obtained in the various results of the previous sections, one may be able to show periodicity or be able to give a bound on i_X that does not depend on X. Unfortunately, there is little chance of this result being true. We studied for all triplet $(c_m, w_d, r_d) \in \{1,\dots,9\}\times\{1,\dots,22\}\times\{1,\dots,24\}$ the value of i_X (by computing most optimal solutions of every value $l \in \{1,\dots,l_X\}$) and reported it in Figure 2.

An important observation from Figure 2 is that (i) most of the time i_X is very small (0 or 1), and (ii) i_X is highly influenced by c_m, for instance when $c_m \geq 2$, we have never obtained values for i_X greater than 10. On the contrary, when $c_m = 1$ and w_d is large, then i_X can take large values (we have observed up to 82). In general, 60% of the time i_X is equal to 0, 93% of the time i_X is less than 3.

5.7.2 Can we compute m_X?

The second question that can be raised is about the value of m_X. Unfortunately we have been unable to prove an analytical formula for m_X, this remains an open question. However by studying values of m_X for small values of c_m, w_d and r_d, we have conjectured an analytical formula for m_X. We have then verified

it on a larger set ($(c_m, w_d, r_d) \in \{1, ..., 10\} \times \{1, ..., 50\} \times \{1, ..., 50\}$) without finding any counter-examples.

Conjecture 1. *We define f a function that takes two integers x, y and c_m as follow:*

$$f : (x, y, c_m) \mapsto \beta(c_m + 1, x + y - 1) - \sum_{k=0}^{y-1} \beta(c_m, k).$$

Let $i_r(r_d, c_m) = x$ or $x + 1$, where x is the only integer that satisfies

$$\beta(c_m + 1, x - 1) \leq r_d < \beta(c_m + 1, x).$$

Let $i_w(w_d, r_d, c_m) = y$, where y is the only integer that satisfies

$$\sum_{j=1}^{y-1} f(j, i_r(r_d, c_m), c_m) < w_d \leq \sum_{j=1}^{i_w} f(j, i_r(r_d, c_m), c_m).$$

We conjecture that

$$m_X = f(i_r(r_d, c_m), i_w(w_d, r_d, c_m), c_m).$$

Note that we have not been able to conjecture the exact value of $i_r(r_d, c_m)$, hence leaving the possibility of two distinct periods m_X.

We give the example below that we used along with the online encyclopedia of integer sequences (OEIS) [7] to be able to do the conjecture.

$c_m = 2$

r_d	m_X					
1	3	9	19	34	55	83
4	6	16	31	52	80	116
10	10	25	46	74		
20	15	36	64	100		

$c_m = 3$

r_d	m_X				
1	4	14	34	69	125
5	10	30	65	121	
15	55	111			

Table 2: Periods when $c_m = 2$ (left) and $c_m = 3$ (right) when w_d increases, for given r_d. When r_d is between two values (for instance for $r_d = 3$), then m_X will vary with w_d between the values from the adjoint sequences ($r_d = 1$ and $r_d = 4$ (resp. 5) for $c_m = 2$ (resp. 3)).

6 Asymptotically optimal online algorithm

We now present an asymptotically optimal online algorithm Disk-A-Revolve to prove Corollary 1. Intuitively, this algorithm writes disk checkpoints periodically with a period of m_X.

Theorem 10. Disk-A-Revolve *is asymptotically optimal for* $\text{Prob}_\infty(l, c_m, w_d, r_d)$.

Algorithm 5 DISK-A-REVOLVE

1: **procedure** DISK-A-REVOLVE(stream)
2: ind $\leftarrow 0$
3: $n_l^o \leftarrow -1$
4: **while** stream is not finished **do**
5: **if** ind $\equiv 0 \mod m_X$ **then**
6: Execute: $D^m_{\text{ind}-m_X} W^d_{\text{ind}} \cdot W^m_{\text{ind}}$ `/*`$D^m_{\text{ind}-m_X}$ `discards the previous checkpoint from memory so that at all time we only store at most one checkpoint in memory.*/`
7: $n_l^o + +$
8: **end if**
9: Execute: F_{ind}
10: ind $+ +$
11: **end while**
12: $l \leftarrow$ ind
13: res $\leftarrow l - n_l^o m_X$
14: Execute: $\bar{F}_l$
15: Execute: $R^m_{l-\text{res}}$SHIFT (REVOLVE(res $-1, c_m$), $n_l^o \cdot m_X$)
16: **for** $i = n_l^o$ downto 1 **do**
17: Execute: $R^d_{i \cdot m_X} \cdot$ SHIFT (1D-REVOLVE($m_X - 1, c_m, r_d$), $i \cdot m_X$)
18: **end for**
19: **return** $\mathcal{S}$
20: **end procedure**

Proof. At the end of the while loop (line 4), $n_l^o = \lfloor l/m_X \rfloor - 1$ (indeed, the "if" test (line 5) is executed $\lfloor l/m_X \rfloor$ times). The execution time of DISK-A-REVOLVE is:

- The while loop (line 4) does $n_l^o + 1$ write to disks and l forward steps.
- We then execute one backward operation, $\bar{F}_l$.
- The execution of REVOLVE then has a cost of $\text{Opt}_0(\text{res} - 1, c_m)$
- Then the for loop (line 16) has a cost of $n_l^o \cdot \left(r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right)$

The total execution time is then:

$$\mathcal{E}xec\,(\text{DISK-A-REVOLVE(stream of size } l)) =$$
$$l u_f + (n_l^o + 1)\, w_d + u_b + \text{Opt}_0(\text{res} - 1, c_m) + n_l^o \cdot \left(r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right)$$

Then, asymptotically,

$$\begin{aligned}\mathcal{E}xec\,(\text{DISK-A-REVOLVE(stream of size } l)) &\sim l u_f + n_l^o \left(w_d + r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right) \\ &\sim l u_f + (\lfloor l/m_X \rfloor - 1) \left(w_d + r_d + \text{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right)\end{aligned}$$

According to Theorems 2 and 5, the optimal execution time is greater than

$$(l - l_X)u_f + \left\lfloor \frac{l - l_X}{m_X} \right\rfloor \cdot \left(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right),$$

which, in turn is equivalent to $lu_f \left(\lfloor l/m_X \rfloor - 1 \right) \left(w_d + r_d + \mathrm{Opt}_1^{(d)}(m_X - 1, c_m, r_d) \right)$ asymptotically. Hence showing that DISK-A-REVOLVE is asymptotically optimal. □

Remark. Note that the execution time of DISK-A-REVOLVE can still be improved by using efficiently the memory checkpoints and waiting until the last minute before storing data on disks. For instance, since m_X is small, we can use in practice the optimal online algorithm designed by Heuveline and Walter [5] for the memory checkpoints between disk checkpoints. However for readability reasons we chose not to present it here as it would not have changed the final result.

7 Evaluation of a periodic schedule

Using Theorem 2, we can easily compute a DOS for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. We need first to pre-compute a DOS for every adjoint graph of size smaller than l_X. Then, we can give in constant time a DOS for any l using the following simple algorithm:

1. Let $n_1 = \left\lceil \frac{l - l_X}{m_X} \right\rceil$. Intuitively, n_1 is the number of periods that we are sure will be equal to m_X.
2. Let $l' = l - n_1 \cdot m_X$. Intuitively, l' is the remainder of the work to be done. A DOS for l' has already been pre-computed: $(m_1, \cdots, m_{n_2}; \mathrm{res})$.
3. Then $(m_X, \ldots, m_X, m_1, \cdots, m_{n_2}; \mathrm{res})$ (with n_1 iterations of m_X initially) is a DOS for l

This gives us a constant time algorithm for $\textsc{Prob}_\infty(l, c_m, w_d, r_d)$. However, the pre-computation part of this algorithm can be costly (in time and space) depending on the parameter of the platform.

7.1 Periodic Dominant Schedules

Based on the observation made in Section 5.7.1, that, in general, i_X is small, we might be interested in only considering the Periodic Dominant Sequence, defined as follows:

Definition 14 (PDS(l, c_m, w_d, r_d))**.** Given values of l, c_m, w_d and r_d, the Periodic Dominant Sequence PDS(l, c_m, w_d, r_d) is the Dominant Sequence

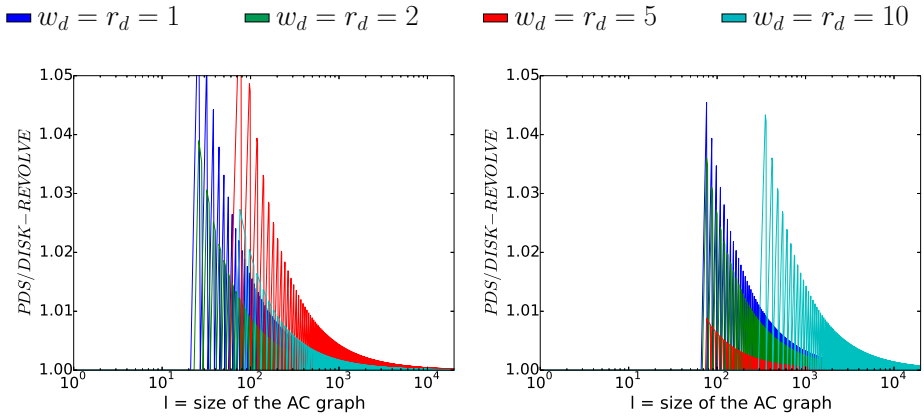
$$\mathrm{PDS}(\mathrm{l},\, c_m,\, w_d,\, r_d) = (m_X, m_X, \cdots, m_X; \mathrm{res})$$

where $\mathrm{res} = \left\lceil \frac{l - l_X^{(1)}}{m_X} \right\rceil$.

The PDS(l, c_m, w_d, r_d) is not always optimal but it has the advantage of not requiring a costly pre-computation. It only needs the value of m_X, $l_X^{(1)}$ and the schedule 1D-REVOLVE (m_X, c_m, r_d).

7.2 Experimental Results

In this Section, we assess the time overhead of the Periodic Dominant Sequence and of DISK-A-REVOLVE compared with the optimal sequence that computes an AC graph of size l. In the experiments, we normalize every time values by setting $u_f = 1$. Because the backward steps are computed exactly once in any solution, their cost has no impact on the algorithms: we set $u_b = 0$ so that we can assess the performance of the algorithms on the forward sweep. Here we present results for $c_m \in \{5, 10\}$ and $w_d = r_d \in \{1, 2, 5, 10\}$.



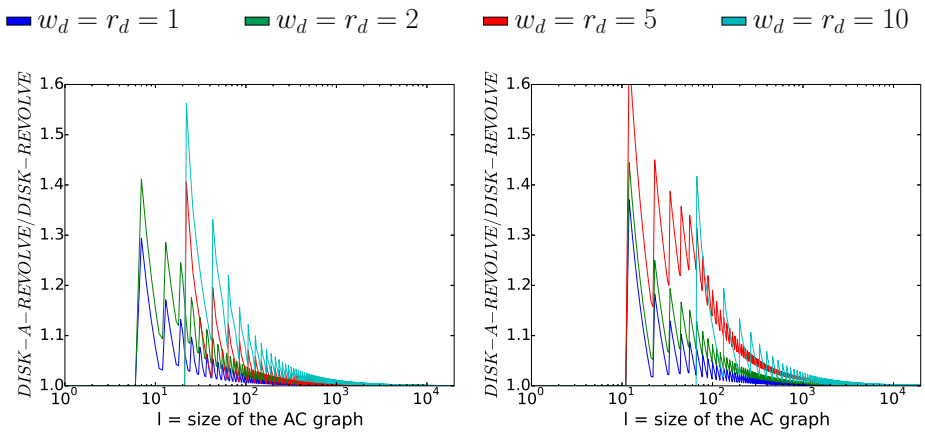
(a) $c_m = 5$ (b) $c_m = 10$

Figure 3: Ratio of the Makespan of PDS(l, c_m, w_d, r_d)/DISK-REVOLVE (l, c_m, w_d, r_d) as a function of l.

Figure 3a and 3b depict the time overhead of PDS(l, c_m, w_d, r_d) compared with DISK-REVOLVE (l, c_m, w_d, r_d) as a function of l. We observe that the ratio increases with w_d and r_d but for large instances of the problem (l greater than 1000), the time overhead is always less than 1%.

Figure 4a and 4b depicts the time overhead of DISK-A-REVOLVE over the optimal sequence DISK-REVOLVE as a function of l. One can see that the asymptotical optimality is attained very quickly (for Adjoint Computation graphs of length 1000). As in the offline case, the ratio increases with w_d and r_d. Note that as it is, the online algorithm is not suited for small instances of the problem (small values of l), indeed, we focused on an asymptotically optimal algorithm. However we expect that mixing our asymptotical algorithm with existing online algorithms [9, 11] could improve those cases.

Overall, both PDS(l, c_m, w_d, r_d) and DISK-A-REVOLVE offer good and ef-



(a) $c_m = 5$ (b) $c_m = 10$

Figure 4: Ratio of the Makespan of Disk-A-Revolve/ Disk-Revolve as a function of l.

ficient algorithms to compute the solution to large Adjoint Computation problems.

8 Conclusion

In this paper, we have designed optimal algorithms for the multi-stage adjoint computation, with two types of storage location: a limited one with no reading and writing costs and an unlimited one with non-zero access times. We improved the time complexity of the optimal dynamic program introduced in our previous work [1]: by showing that the optimal solution is weakly periodic, we were able to construct an algorithm that returns an optimal schedule for the problem with constant overhead (compared to the quadratic time-complexity of the optimal algorithm presented in [1]). Furthermore, we also developed asymptotic optimal algorithms that need almost no precomputation. Finally, we provided an asymptotical optimal algorithm for the online problem (when the graph size is not known before-hand).

Modern large-scale cluster are subject to failures that can occur at any time during the computation and lead to a memory flush. The algorithm introduced in this paper establishes a solid foundation to study the impact of memory failures on the performance of multi-stage adjoint computation.

Acknowledgements

The authors would like to thank Paul Hovland for introducing them to the problem and the idea of a periodic schedule. They would further like to thank

Yves Robert and Paul Hovland for constructive conversations. This material is based upon work supported by the French Research Agency (ANR) through the Rescue project and the Joint Laboratory for Extreme-Scale Computing

References

[1] G. Aupy, J. Herrmann, P. Hovland, and Y. Robert, *Optimal multistage algorithm for adjoint computation*, Research report RR-8721, INRIA, Apr. 2015, available at `http://gaupy.org`.

[2] R. Giering and T. Kaminski, *Recomputations in reverse mode AD*, in *Automatic Differentiation: From Simulation to Optimization*, G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, eds., chap. 33, Computer and Information Science, Springer, New York, 2002, pp. 283–291, Available at `http://www.springer.de/cgi-bin/search_book.pl?isbn=0-387-95305-1`.

[3] A. Griewank, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optimization Methods and software 1 (1992), pp. 35–54.

[4] A. Griewank and A. Walther, *Algorithm 799: Revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation*, ACM Transactions on Mathematical Software (TOMS) 26 (2000), pp. 19–45.

[5] V. Heuveline and A. Walther, *Online checkpointing for parallel adjoint computation in pdes: Application to goal-oriented adaptivity and flow control*, in *Euro-Par 2006 Parallel Processing*, Springer, 2006, pp. 689–699.

[6] R.R. Phelps, *Convex functions, monotone operators and differentiability*, 1364, Springer Science & Business Media, 1993.

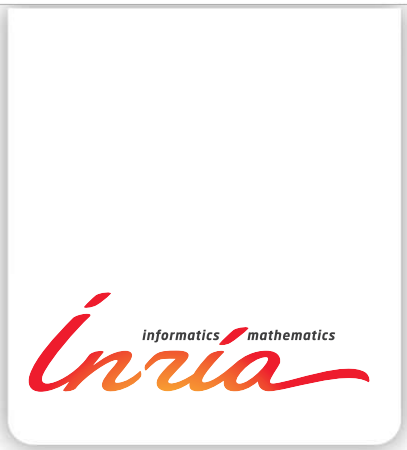
[7] N.O. SLOANE, *Online encyclopaedia of integer sequences*, Published electronically at http://www.oeis.org. Accessed July 2015. (2015).

[8] P. Stumm and A. Walther, *Multistage approaches for optimal offline checkpointing*, SIAM Journal on Scientific Computing 31 (2009), pp. 1946–1967.

[9] P. Stumm and A. Walther, *New algorithms for optimal online checkpointing*, SIAM Journal on Scientific Computing 32 (2010), pp. 836–854.

[10] A. Walther, *Program reversal schedules for single-and multi-processor machines*, Ph.D. diss., PhD thesis, Institute of Scientific Computing, Technical University Dresden, Germany, 1999.

[11] Q. Wang, P. Moin, and G. Iaccarino, *Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation*, SIAM Journal on Scientific Computing 31 (2009), pp. 2549–2567.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399