Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format

Patrick Amestoy, Alfredo Buttari, Jean-Yves L'Excellent, Théo Mary

Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format

Patrick R. Amestoy* Alfredo Buttari† Jean-Yves L'Excellent‡
Theo Mary§

Abstract

Matrices possessing a low-rank property arise in numerous scientific applications. This property can be exploited to provide a substantial reduction of the complexity of their LU or LDL^T factorization. Among the possible low-rank formats, the flat Block Low-Rank (BLR) format is easy to use but achieves superlinear complexity. Alternatively, the hierarchical formats achieve linear complexity at the price of a much more complex, hierarchical matrix representation. In this paper, we propose a new format based on multilevel BLR approximations: the matrix is recursively defined as a BLR matrix whose full-rank blocks are themselves represented by BLR matrices. We call this format *multilevel BLR* (MBLR). Contrarily to hierarchical matrices, the number of levels in the block hierarchy is fixed to a given constant; while this format can still be represented within the $\mathscr{H}$ formalism, we show that applying the $\mathscr{H}$ theory to it leads to very pessimistic complexity bounds. We therefore extend the theory to prove better bounds, and show that the MBLR format provides a simple way to finely control the desired complexity of dense factorizations. By striking a balance between the simplicity of the BLR format and the low complexity of the hierarchical ones, the MBLR format bridges the gap between flat and hierarchical low-rank matrix formats. The MBLR format is of particular relevance in the context of sparse direct solvers, for which it is able to trade off the optimal dense complexity of the hierarchical formats to benefit from the simplicity and flexibility of the BLR format while still achieving $O(n)$ sparse complexity. We finally compare our MBLR format with the related BLR-$\mathscr{H}$ (or Lattice-$\mathscr{H}$) format; our theoretical analysis shows that both formats achieve the same asymptotic complexity for a given top level block size.

1 Introduction

Efficiently computing the solution of a dense linear system is a fundamental building block of numerous scientific computing applications. Let us refer to such a system as

$$Fu_F = v_F, \tag{1}$$

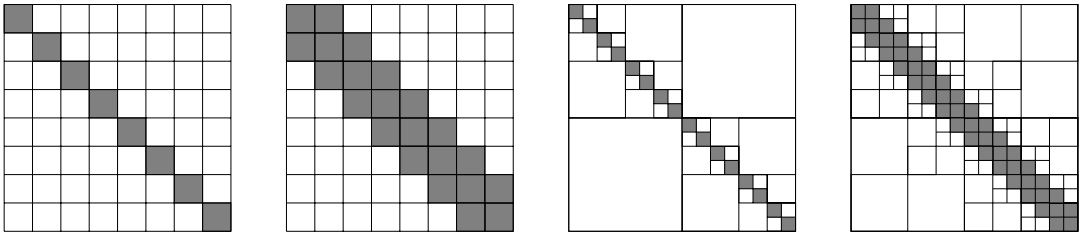
where F is a dense matrix of order m, u_F is the unknown vector of size m, and v_F is the right-hand side vector of size m.

This paper focuses on solving (1) with direct approaches based on Gaussian elimination, which consist in factorizing matrix F as $F = LU$ or $F = LDL^T$, depending on whether the matrix is unsymmetric or symmetric, respectively.

In many applications (e.g., Schur complements arising from the discretization of elliptic partial differential equations), the matrix F has been shown to have a low-rank property: conveniently defined off-diagonal blocks can be approximated by low-rank matrices [10].

Several formats have been proposed to exploit this property. The simplest one is the Block Low-Rank (BLR) format [2], which partitions the matrix with a flat, 2D blocking and approximates

*University of Toulouse, INP-IRIT
†University of Toulouse, CNRS-IRIT
‡University of Lyon, CNRS, ENS de Lyon, Inria, UCBL, LIP UMR5668, France
§University of Manchester



(a) BLR partitioning (weak admissibility).
(b) BLR partitioning (strong admissibility).
(c) HODLR partitioning (weak admissibility)
(d) $\mathscr{H}$ partitioning (strong admissibility)

Figure 1: Illustration of different low-rank formats. Gray blocks are stored in full-rank whereas white ones are approximated by low-rank matrices.

its off-diagonal blocks by low-rank submatrices, as illustrated in Figure 1(a). Compared with the cubic $O(m^3)$ complexity of the dense full-rank LU or LDL^T factorizations, the complexity of the dense BLR factorization can be as low as $O(m^2)$ [4].

More advanced formats are based on a hierarchical partitioning of the matrix: the matrix F is partitioned with a 2×2 blocking and the two diagonal blocks are recursively refined, as illustrated in Figures 1(c) and 1(d). Different hierarchical formats can be defined depending on whether the off-diagonal blocks are directly approximated (so-called *weakly-admissible* formats) or further refined (so-called *strongly-admissible* formats). The most general of the hierarchical formats is the strongly-admissible $\mathscr{H}$-matrix format [10, 20, 11]; the HODLR format [6] is its weakly-admissible counterpart. These hierarchical formats can factorize a dense matrix in near-linear complexity $O(m \log^q m)$, where q is a small integer that depends on which factorization algorithm is used; in the following, we will consider $q = 2$. The log factor can be removed by using a so-called *nested-basis* structure. The strongly-admissible $\mathscr{H}^2$-matrix format [11] and the weakly-admissible HSS [30, 12] and HBS [18] formats exploit such nested basis structures to achieve linear complexity $O(m)$.

In this paper, we propose a new format based on multilevel BLR approximations. The matrix F is recursively represented as a BLR matrix whose full-rank blocks are themselves BLR matrices. We call this format *multilevel BLR* (MBLR). In the hierarchical format, the matrix is refined until the diagonal blocks are of constant size; this therefore leads to a number of levels in the block hierarchy which is nonconstant (usually $O(\log_2 m)$). With the MBLR format, we propose to make this number of levels a tunable parameter to be set to a given value ℓ that does not asymptotically depend on the matrix size m. We prove that this parameter provides a simple way to finely control the complexity of the dense MBLR factorization. The complexity varies from $O(m^2)$ for monolevel BLR down to nearly $O(m)$ for an infinite number of levels. By striking a balance between the simplicity of the BLR format and the low complexity of the hierarchical ones, the MBLR format bridges the gap between flat and hierarchical low-rank matrix formats.

We will show that the MBLR format is of particular relevance in the context of sparse direct solvers, which aim to compute the solution of a sparse linear system

$$Au_A = v_A, \tag{2}$$

where A is a sparse matrix of order n, u_A is the unknown vector of size n, and v_A is the right-hand side vector of size n. Two widely studied classes of sparse direct methods are the multifrontal [14, 23] and supernodal [9, 13] approaches.

Sparse direct methods rely on a sequence of partial factorizations of dense matrices F, referred to as supernodes or fronts. Therefore, the complexity of sparse direct methods is directly derived from the complexity of the factorization of each dense matrix F. For example, with an adequate reordering, a well-known result is that the dense standard full-rank $O(m^3)$ factorization leads to a $O(n^2)$ sparse complexity for regular 3D problems [15]. The low-rank formats described above can be efficiently exploited within sparse solvers to provide a substantial reduction of their complexity.

The potential of BLR sparse solvers has been first investigated in [2]; the simplicity and flexibility of the BLR format makes it easy to use in the context of a general purpose, algebraic solver, as presented in [5, 3, 27, 25]. [5] focuses on the multicore performance of BLR multifrontal solvers, while [3] and [27] present their use in two real-life industrial applications coming from geosciences. [25] present the use of the BLR format in supernodal solvers. Furthermore, it has been proved in [4] that the theoretical complexity of the BLR multifrontal factorization may be as low as $O(n^{4/3})$ (for 3D problems with constant ranks).

Alternatively, most sparse solvers based on the more complex hierarchical formats have been shown to possess near-linear complexity. To cite a few, [29, 28, 17, 16] are HSS-based, [18] is HBS-based, [7] is HODLR-based, and [26] is $\mathscr{H}^2$-based.

However, a critical observation is that achieving $O(n)$ sparse complexity does not actually require a linear dense complexity $O(m)$. For instance, for 3D problems, all that is required is a dense complexity lower than $O(m^{1.5})$. Therefore, we will prove that the MBLR format is able to trade off the optimal dense complexity of the hierarchical formats to benefit from the simplicity and flexibility of the flat BLR format while still achieving $O(n)$ sparse complexity.

We now describe the organization of the rest of this paper. In Section 2, we provide some background on the BLR factorization and its complexity and we motivate the key idea behind MBLR approximations. We explain in Section 3 how the MBLR format can be described using the cluster tree representation commonly used in the $\mathscr{H}$ literature; this provides a convenient way to explain the key difference between the MBLR and $\mathscr{H}$ formats. We show that, similarly to the BLR case, the $\mathscr{H}$ theoretical formalism leads to MBLR complexity bounds that are very pessimistic. We therefore extend the theory, beginning by the two-level case in Section 4. We prove that two levels can already significantly improve the theoretical complexity of the factorization. In Section 5, we generalize the previous proof to the MBLR format with an arbitrary number of levels; we prove that, for constant ranks, only four levels are already enough to reach $O(n)$ sparse 3D complexity (and three levels already achieve near-linear $O(n \log n)$ complexity). In Section 6, we validate our theoretical results with numerical experiments. In Section 7, we compare our MBLR format to a related format, the BLR-$\mathscr{H}$ format (also referred to as "Lattice-$\mathscr{H}$"). We provide our concluding remarks in Section 8. In the main body of this article, we consider for the sake of simplicity the weakly-admissible case, in which only the diagonal blocks are refined. In the appendix, we provide the extension of the MBLR format to the strongly-admissible case, in which off-diagonal full-rank blocks are also recursively refined. We prove that our complexity bounds remain valid in this context.

2 Background and motivation

2.1 Block Low-Rank approximations

The BLR format is based on a flat, non-hierarchical blocking of the matrix which is defined by conveniently clustering the associated unknowns. A BLR representation $\widetilde{F}$ of a dense matrix F is shown in (3), where we assume that $p \times p$ blocks have been defined. Off-diagonal blocks F_{ij} ($i \neq j$) of size $m_i \times n_j$ and numerical rank k_{ij}^{ε} are approximated by a low-rank matrix $\widetilde{F}_{ij} = X_{ij} Y_{ij}^T$ at accuracy ε, where X_{ij} is a $m_i \times k_{ij}^{\varepsilon}$ matrix and Y_{ij} is a $n_j \times k_{ij}^{\varepsilon}$ matrix. The diagonal blocks F_{ii} are stored as full-rank matrices ($\widetilde{F}_{ii} = F_{ii}$).

$$\widetilde{F} = \begin{bmatrix} \widetilde{F}_{11} & \widetilde{F}_{12} & \cdots & \widetilde{F}_{1p} \\ \widetilde{F}_{21} & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \widetilde{F}_{p1} & \cdots & \cdots & \widetilde{F}_{pp} \end{bmatrix}. \tag{3}$$

Throughout this article, we will note F_{ij} the (i,j)-th block of F and $F_{:,k}$ its k-th block-column. We will also assume that all blocks have a size of order b, i.e., $m_i = n_j = O(b)$.

Computing the low-rank approximation $\widetilde{F}_{ij}$ to each block, referred to as the *compression* step, can be performed in different ways. We have chosen to use a truncated QR factorization with

column pivoting; this corresponds to a QR factorization with pivoting which is truncated as soon as a diagonal coefficient of the R factor falls below the prescribed threshold ε. For a block of size $b \times b$ and rank r, the cost of the compression is $O(b^2 r)$, whereas computing the exact singular value decomposition of the block would require $O(b^3)$ operations. This choice thus allows for a convenient compromise between cost and accuracy of the compression operation.

2.2 Block Low-Rank dense factorization

We describe in Algorithm 1 the CUFS variant of the BLR factorization algorithm for dense matrices, introduced in [4]. Algorithm 1 is presented in its LU version, but it can easily be adapted to the symmetric case.

In order to perform the LU or LDL^T factorization of a dense BLR matrix, the standard block LU or LDL^T factorization has to be modified so that the low-rank blocks can be exploited to perform fast operations. Many such algorithms can be defined depending on where the compression step is performed. As described in [4], the CUFS variant achieves the lowest complexity of all BLR variants by performing the compression as early as possible.

Algorithm 1 Dense BLR LU factorization: CUFS variant.

Input: a $p \times p$ block matrix F of order m.
Output: F overwritten by its BLR LU factors $\widetilde{F}$.

for $k = 1$ **to** p **do**
  for $i = k+1$ **to** p **do**
    Compress (L): $F_{ik} \leftarrow \widetilde{F}_{ik} = X_{ik} Y_{ik}^T$
    Compress (U): $F_{ki} \leftarrow \widetilde{F}_{ki} = Y_{ki} X_{ki}^T$
  end for
  for $i = k$ **to** p **do**
    for $j = 1$ **to** $k-1$ **do**
      Update (L): $\widetilde{F}_{ik} \leftarrow \widetilde{F}_{ik} - X_{ij} Y_{ij}^T Y_{jk} X_{jk}^T$
      Update (U): $\widetilde{F}_{ki} \leftarrow \widetilde{F}_{ki} - X_{kj} Y_{kj}^T Y_{ji} X_{ji}^T$
    end for
    $\widetilde{F}_{ik} \leftarrow \mathrm{Recompress}(\widetilde{F}_{ik})$
    $\widetilde{F}_{ki} \leftarrow \mathrm{Recompress}(\widetilde{F}_{ki})$
  end for
  Factor: $F_{kk} = L_{kk} U_{kk}$
  for $i = k+1$ **to** p **do**
    Solve (L): $\widetilde{F}_{ik} \leftarrow \widetilde{F}_{ik} U_{kk}^{-1} = X_{ik} Y_{ik}^T U_{kk}^{-1}$
    Solve (U): $\widetilde{F}_{ki} \leftarrow L_{kk}^{-1} \widetilde{F}_{ki} = L_{kk}^{-1} X_{ki} Y_{ki}^T$
  end for
end for

This algorithm is referred to as CUFS (standing for Compress, Update, Factor, and Solve), to indicate the order in which the steps are performed. All low-rank updates of a given block $\widetilde{F}_{ik}$ are accumulated together before being recompressed, in order to achieve the smallest rank possible for $\widetilde{F}_{ik}$.

The CUFS BLR variant is referred to as *fully-structured*, which means the off-diagonal low-rank blocks $\widetilde{F}_{ik}$ are never stored in full-rank again after being initially compressed. Furthermore, in the rest of this article, we will assume that the matrix is already available under compressed form, i.e., the Compress step is free or at least it can be performed at a cost that is small enough (e.g., by means of a fast matrix-vector multiply such as SpMV) so that it is negligible with regard to the total complexity of the factorization.

One of the main results of [4] is that the storage complexity of the factorization of a dense matrix of order m with off-diagonal blocks of rank at most r is equal to

$$\mathcal{S}_{ds}^1(m,r) = O(m^{1.5}\sqrt{r}) \tag{4}$$

Table 1: Flop and storage complexities of the factorization of a sparse system of $n = N \times N$ (2D case) or $n = N \times N \times N$ (3D case) unknowns, assuming a dense factorization complexity $O(m^\beta)$.

2D		3D	
$\beta > 2$	$O(n^{\beta/2})$	$\beta > 1.5$	$O(n^{2\beta/3})$
$\beta = 2$	$O(n \log n)$	$\beta = 1.5$	$O(n \log n)$
$\beta < 2$	$O(n)$	$\beta < 1.5$	$O(n)$

and the flop complexity is

$$\mathscr{F}^1_{ds}(m,r) = O(m^2 r). \tag{5}$$

The proof of this result will be recalled in Section 2.4. The 1 superscript refers to the monolevel BLR factorization. This notation will be generalized in the next sections.

2.3 BLR sparse factorization

Because sparse direct factorizations such as multifrontal or supernodal approaches rely on dense factorizations, block low-rank approximations can easily be incorporated into the sparse factorization by representing the fronts (or supernodes) with the chosen low-rank format. For example, in the BLR case, the fronts are represented as defined by (3), and Algorithm 1 is adapted to perform their partial factorization. This is described in detail in [2, 24].

As a consequence, the complexity of the sparse factorization can be directly computed from the complexity of the dense factorization. We consider a matrix of order $n = N^d$, where d denotes the problem dimension, that is reordered using nested dissection [15]: the domain is recursively partitioned by so-called *separators*. The sparse complexity is then computed as follows (assuming cross-shaped separators): at each level ℓ of the separator tree, we need to factorize $(2^d)^\ell$ fronts of order $O((N/2^\ell)^{d-1})$, for ℓ ranging from 0 to $L = \log_2(N)$. Therefore, the flop complexity $\mathscr{F}_{sp}(N)$ is equal to

$$\mathscr{F}_{sp}(N) = \sum_{\ell=0}^{L} (2^d)^\ell \mathscr{F}_{ds}((\frac{N}{2^\ell})^{d-1}), \tag{6}$$

where $\mathscr{F}_{ds}(m)$ is the dense flop complexity. In BLR, it is given by (5). Similarly, the storage complexity $\mathscr{S}_{sp}(N)$ is equal to

$$\mathscr{S}_{sp}(N) = \sum_{\ell=0}^{L} (2^d)^\ell \mathscr{S}_{ds}((\frac{N}{2^\ell})^{d-1}), \tag{7}$$

where $\mathscr{S}_{ds}(m)$ is the dense storage complexity. In BLR, it is given by (4).

Assuming a dense complexity $O(m^\beta)$, it can easily be shown from (6) and (7) that the sparse complexities only depend on the dense complexity exponent β and the dimension d. This correspondence is reported in Table 1. The key observation is that *a linear $O(m)$ dense complexity is not required to achieve a linear $O(n)$ sparse complexity*. In fact, a dense complexity lower than $O(m^2)$ and $O(m^{1.5})$ suffices for 2D and 3D problems, respectively.

Then, assuming a constant rank bound $r = O(1)$, the sparse complexities are reported in Table 2, for the FR, BLR, and $\mathscr{H}$ factorizations. Thanks to the key observation above, the BLR sparse complexities are not that far from the $\mathscr{H}$ complexities. For example, the BLR 2D storage complexity is already optimal; furthermore, the 2D flop and 3D storage complexities are nearly linear, with only an additional $O(\log n)$ factor compared with $\mathscr{H}$. More importantly, thanks to the same key observation, we only need a modest improvement of the dense complexity to reach $O(n)$ complexity. Specifically:

- To drop the $O(\log n)$ factor in the 2D flop complexity, the dense complexity $\mathscr{F}_{ds}(m)$ only needs to be strictly inferior to $O(m^2)$;
- Similarly, to drop the $O(\log n)$ factor in the 3D storage complexity, the dense complexity $\mathscr{S}_{ds}(m)$ only needs to be strictly inferior to $O(m^{1.5})$;

Table 2: Flop and storage complexities of the FR, BLR, and $\mathscr{H}$ factorizations of a sparse system of $n = N \times N$ (2D case) or $n = N \times N \times N$ (3D case) unknowns, derived from the complexities of the factorization of a dense matrix of order m. We consider a constant rank bound $r = O(1)$. The BLR variant considered is CUFS.

	$\mathscr{F}_{ds}(m)$	$\mathscr{F}_{sp}(n)$		$\mathscr{S}_{ds}(m)$	$\mathscr{S}_{sp}(n)$	
		2D	3D		2D	3D
FR	$O(m^3)$	$O(n^{1.5})$	$O(n^2)$	$O(m^2)$	$O(n \log n)$	$O(n^{4/3})$
BLR	$O(m^2)$	$O(n \log n)$	$O(n^{4/3})$	$O(m^{1.5})$	$O(n)$	$O(n \log n)$
$\mathscr{H}$	$O(m \log^2 m)$	$O(n)$	$O(n)$	$O(m \log m)$	$O(n)$	$O(n)$

- Finally, the superlinear 3D flop complexity can be made linear with a dense complexity $\mathscr{F}_{ds}(m)$ strictly inferior to $O(m^{1.5})$.

The main motivation behind the MBLR format is to find a *simple* modification of the BLR factorization, that preserves its simplicity and flexibility, while improving the complexity *just enough* to get the desired exponent. We will prove in Section 5 that this complexity improvement can be controlled by the number of levels.

2.4 Complexity analysis for BLR

To motivate the key idea behind MBLR, let us summarize the dense storage and flop complexity analysis found in [4] that leads to the formulas (4) and (5). We consider the CUFS variant of the BLR factorization. As indicated in the introduction, we first consider the weakly-admissible case: we assume that all off-diagonal blocks are low-rank. The extension to the strongly-admissible case is provided in the appendix.

We consider a dense BLR matrix of order m. We note b the block size and $p = m/b$ the number of blocks per row and/or column. The amount of storage required to store the factors of such a matrix can be computed as the sum of the storage for the full-rank diagonal blocks and that for the low-rank off-diagonal blocks:

$$\mathscr{S}^1_{ds}(b,p,r) = O(pb^2) + O(p^2br). \tag{8}$$

Then, we assume that the block size b is of order $O(m^x)$, where x is a real value in $[0,1]$, and thus the number of blocks p per row and/or column is of order $O(m^{1-x})$. We also assume that the rank bound is of the form $r = O(m^\alpha)$. By replacing b, p, and r by their expression in (8), we obtain an expression of $\mathscr{S}^1_{ds}$ which depends on (m,x,α) instead of (b,p,r):

$$\mathscr{S}^1_{ds}(m,x,\alpha) = O(m^{1+x}) + O(m^{2-x+\alpha}). \tag{9}$$

We then define x^* as the optimal choice of x which minimizes the asymptotic complexity of (9). x^* can be computed as the value which makes each term in (9) asymptotically equal. We obtain

$$x^* = (1+\alpha)/2, \tag{10}$$

which means the optimal choice of block size is

$$b^* = O(\sqrt{mr}). \tag{11}$$

This leads to the final dense complexity (already given in (4))

$$\mathscr{S}^1_{ds}(m,r) = O(m^{1.5}\sqrt{r}). \tag{12}$$

Next, to compute the flop complexity, we compute the cost of the Factor, Solve, Product, and Recompress steps and report them in Table 3 (third column). This cost depends on the type (full-rank or low-rank) of the block(s) on which the operation is performed (second column). Note that

Table 3: Main operations for the BLR factorization of a dense matrix of order m, with blocks of size b, and low-rank blocks of rank at most r. We note $p = m/b$. Type: type of the block(s) on which the operation is performed. $cost^1_{Step}$: cost of performing the operation once. $number_{Step}$: number of times the operation is performed. $\mathscr{F}^1_{Step}$: obtained by multiplying the $cost^1_{Step}$ and $number_{Step}$ columns (equation (13)). The first expression is given as a function of b, p, and r, while the second is obtained with the assumption that $b = O(m^x)$ (and thus $p = O(m^{1-x})$) and $r = O(m^\alpha)$, for $x, \alpha \in [0,1]$.

Step	Type	$cost^1_{Step}$	$number_{Step}$	$\mathscr{F}^1_{Step}(b,p,r)$	=	$\mathscr{F}^1_{Step}(m,x,\alpha)$
Factor	FR	$O(b^3)$	$O(p)$	$O(pb^3)$	=	$O(m^{1+2x})$
Solve	FR-LR	$O(b^2r)$	$O(p^2)$	$O(p^2b^2r)$	=	$O(m^{2+\alpha})$
Product	LR-LR	$O(br^2)$	$O(p^3)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$
Recompress	LR	$O(bpr^2)$	$O(p^2)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$

the Product operation can only take the form of a product of two low-rank blocks (LR-LR), because it involves only off-diagonal blocks, which are low-rank in the weakly-admissible case. We also remind that we have assumed that the matrix is already available under compressed form and thus we do not report the Compress step in Table 3.

In the weakly-admissible case, there is only one full-rank block on each block-row (the diagonal one); therefore, we can easily count the number of blocks on which each step is performed; we report it on the fourth column of Table 3. The BLR factorization cost of each step is then equal to

$$\mathscr{F}^1_{Step} = cost^1_{Step} \times number_{Step} \tag{13}$$

and is reported in the fifth and sixth columns of Table 3. In the fifth column, its expression depends on b, p, and r, while in the sixth column it is given as a function of m, x, and α by by substituting b, p, and r by $O(m^x)$, $O(m^{1-x})$, and $O(m^\alpha)$, respectively.

The total flop complexity of the dense BLR factorization is equal to the sum of the cost of all steps

$$\mathscr{F}^1_{ds}(m,x,\alpha) = O(m^{1+2x} + m^{2+\alpha} + m^{3-2x+2\alpha}). \tag{14}$$

We compute x^*, the optimal choice of x which minimizes the complexity, and find again $x^* = (1+\alpha)/2$, which means that the same x^* value minimizes both the storage and flop complexities, a valuable property. We finally obtain

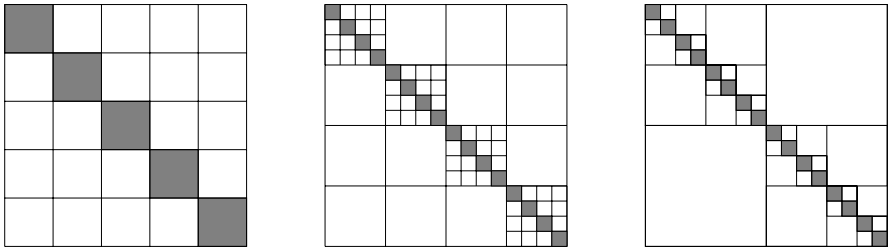
$$\mathscr{F}^1_{ds}(m,r) = O(m^2 r). \tag{15}$$

2.5 Key idea of the MBLR format

Let us now consider each step of Table 3 with the objective of reducing the total cost of the factorization. The Product and Recompress steps involve exclusively low-rank blocks and their cost is already optimal as it is linear with respect to the block size b. Therefore, we focus on the Factor and Solve steps. These steps have superlinear cost with respect to b because they involve the diagonal full-rank blocks.

Thus, the key idea of the MBLR format is to further refine these full-rank diagonal blocks by replacing them by BLR matrices. This is illustrated in Figure 2(b). Compared with the BLR format (Figure 2(a)), we will show in Section 4 that this more compressed representation decreases the cost of performing the Factor and Solve steps, which allows to increase the block size to achieve a lower complexity. However, it also remains very different from a hierarchical format (Figure 2(c)). First, the diagonal blocks are represented by the simple BLR format, rather than a more complex hierarchical format. Second, while larger than in the BLR case, the off-diagonal blocks are in general still much smaller than in the $\mathscr{H}$ case. This has several advantages:

- No relative order is needed between blocks; this allows the clustering to easily be computed and delivers a great flexibility to distribute the data in a parallel environment.



(a) Weakly-admissible, monolevel BLR partitioning.

(b) Weakly-admissible, two-level BLR partitioning.

(c) Weakly-admissible, HODLR partitioning

Figure 2: Comparison of BLR, MBLR, and hierarchical formats in the weakly-admissible case.

- The size of the blocks can be small enough to fit on a single shared-memory node; therefore, in a parallel environment, each processor can efficiently and concurrently work on different blocks.
- To perform numerical pivoting, the quality of the pivot candidates can be estimated with the entries of the low-rank basis of the off-diagonal blocks, as explained in [24]. However, the quality of this estimation depends on the size of the block; smaller blocks lead to a tighter estimate. This makes the BLR format particularly suitable to handle numerical pivoting, a critical feature lacking in most hierarchical solvers presented in the literature.

While many advantages of the BLR and MBLR formats lie in their efficiency and flexibility in the context of a parallel execution, the parallel implementation of the MBLR format is out of the scope of this paper. Similarly, we omit a detailed description of the algorithmics necessary to handle numerical pivoting; see [24] for a thorough discussion. In this paper, we focus on the theoretical complexity analysis of the MBLR factorization.

3 Difference between MBLR and $\mathscr{H}$ matrices

In this section, we first explain how MBLR matrices can be represented using the cluster tree modelization typically used in the $\mathscr{H}$ literature; this provides a convenient tool to formalize the key difference between the MBLR and hierarchical formats that was informally presented in the previous section: the number of levels in the cluster tree is a constant in the MBLR format, while it is logarithmically dependent on the problem size in the $\mathscr{H}$ format.

Using this formalism, $\mathscr{H}$ theory is thus applicable to the MBLR format; however, we show in Section 3.3 that it leads to very pessimistic complexity bounds. We must therefore develop a new theory to compute satisfying bounds, which is the object of Sections 4 and 5. Since the proofs and computations in these sections are *not* based on the $\mathscr{H}$ theory, this section may be skipped by the reader, at least on a first read.

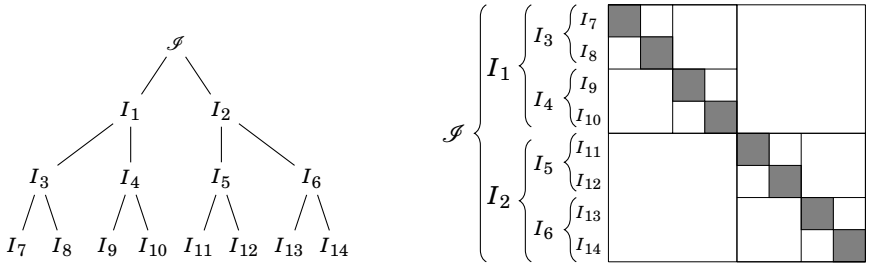
3.1 The hierarchical case

Here, we briefly remind the definition of cluster trees, and how they are used to represent hierarchical partitionings. We refer to [10, 21] for a formal and detailed presentation.

Let us note $\mathscr{I}$ the set of unknowns. We assume that the sets of row and column indices of the matrix are the same, for the sake of simplicity, and because we do not need to distinguish them for the purpose of this section.

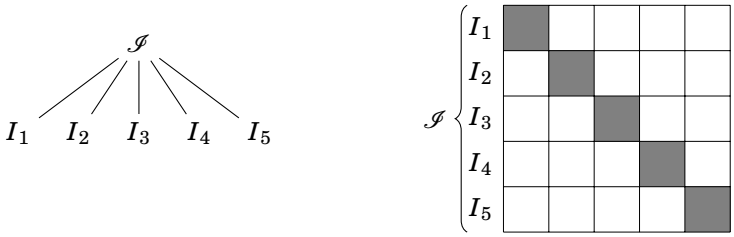
Computing a recursive partition $\mathfrak{S}(\mathscr{I})$ of $\mathscr{I}$ can be modeled with a so-called *cluster tree*.

Definition 1 (Cluster tree)**.** *Let $\mathscr{I}$ be a set of unknowns and $T_{\mathscr{I}}$ a tree whose nodes v are associated with subsets σ_v of $\mathscr{I}$. $T_{\mathscr{I}}$ is said to be a cluster tree iff:*



(a) Cluster tree. (b) HODLR block-clustering.

Figure 3: An example of cluster tree and its associated HODLR block-clustering.



(a) Flat cluster tree. (b) BLR block-clustering.

Figure 4: An example of flat cluster tree and its associated BLR block-clustering.

- *The root of $T_{\mathcal{I}}$, noted r, is associated with $\sigma_r = \mathcal{I}$;*
- *For each non-leaf node $v \in T_{\mathcal{I}}$, with children noted $C_{T_{\mathcal{I}}}(v)$, the subsets associated with the children form a partition of σ_v, i.e.,*

$$\bigcup_{c \in C_{T_{\mathcal{I}}}(v)} \sigma_c = \sigma_v.$$

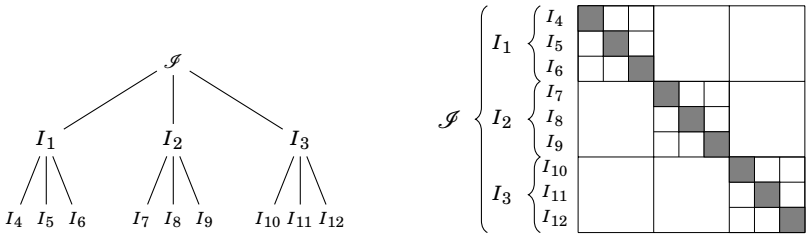
An example of cluster tree is provided in Figure 3(a). As illustrated, cluster trees establish a *hierarchy* between clusters. A given cluster tree uniquely defines a weakly-admissible, hierarchical block-clustering (so-called HODLR), as illustrated in Figure 3(b).

Note that the cluster tree is not enough to define general, strongly-admissible hierarchical block-clusterings such as an $\mathcal{H}$ block-clustering. In that case, a new tree structure, so-called block-cluster tree, must be introduced. For the sake of simplicity, we do not discuss them here, since cluster trees (and weakly-admissible hierarchical block-clusterings) are enough to explain the difference between MBLR and hierarchical matrices.

3.2 The BLR and MBLR formats, explained with cluster trees

It is interesting to first mention how the BLR format can be viewed as a very particular kind of $\mathcal{H}$-matrix. In [4], we made a first attempt to model BLR partitionings with cluster trees, where the BLR partitioning is defined using only the leaves of the cluster tree. However, this model is inadequate because the other nodes of the tree are unnecessary (and thus there are several possible cluster trees for a unique BLR partitioning).

Therefore, we newly propose to define a given BLR partitioning with a unique, flat cluster tree of depth 1, as illustrated in Figure 4.



(a) Cluster tree of depth 2.

(b) Two-level BLR block-clustering.

Figure 5: An example of cluster tree of depth 2 and its associated two-level BLR block-clustering.

With this definition, it is straightforward to extend the model to the MBLR format. An ℓ-level BLR block-clustering can be represented by a cluster tree of depth ℓ. This is illustrated in Figure 5 in the two-level case.

It is thus clear that, just like the BLR format, the MBLR one can also be viewed as a very particular kind of $\mathscr{H}$-matrix, since any MBLR matrix can be represented with a cluster tree that satisfies Definition 1. However, while $\mathscr{H}$-matrices are indeed also represented by cluster trees, in practice, they are virtually always built with an implicit assumption: the number of children of any node in the cluster tree is constant with respect to the size of the matrix. Most commonly, cluster trees are binary trees, and thus this number is 2. Since the diagonal blocks of an $\mathscr{H}$-matrix are of constant size, this leads to $O(\log m)$ levels in the cluster tree.

The MBLR format is based on the converse approach: the number of levels is fixed to some constant $\ell = O(1)$, which in turn leads to a number of children per node that is asymptotically dependent on m. For example, assuming a constant rank $r = O(1)$, we will prove in Section 4 that a two-level BLR matrix is represented by a cluster tree with $O(m^{1/3})$ nodes on the first level (children of the root node), and each of these nodes has $O((m^{2/3})^{1/2}) = O(m^{1/3})$ children on the second level.

While this is technically not prohibited by the general $\mathscr{H}$-format definition, it has, to the best of our knowledge, never been considered in the $\mathscr{H}$ literature. As a matter of fact, in his recent book, Hackbusch writes ([21], p.83): 'The partition should contain as few blocks as possible since the storage cost increases with the number of blocks". While that is true, we believe that the smaller size and greater number of blocks can provide a gain in flexibility that can be useful in a parallel solver.

One may in fact find it surprising that this simple idea of having a constant number of levels has never been proposed before. We believe that this is mainly explained by the fact that the $\mathscr{H}$ theoretical formalism does not provide a satisfying result if applied to the MBLR format, as explained in the following.

3.3 $\mathscr{H}$ theory leads to pessimistic MBLR complexity bounds

In [4, Section 3.2], we explained that applying the $\mathscr{H}$ theoretical complexity formulas to the BLR format does not provide a satisfying complexity bound. Here, we show this remains the case for the MBLR format.

The storage and flop complexities of the factorization of a dense $\mathscr{H}$-matrix of order m have been shown [19] to be

$$\mathscr{S}_{ds}^{\mathscr{H}}(m) = O(mLc_{sp}\max(r_{max}, b_{diag})), \tag{16}$$

$$\mathscr{F}_{ds}^{\mathscr{H}}(m) = O(mL^2c_{sp}^3\max(r_{max}, b_{diag})^2). \tag{17}$$

L is the depth of the cluster tree. c_{sp} is the so-called sparsity constant, defined as the maximum number of blocks of a given level in the cluster tree that are in the same row or column of the matrix; it is a measure of how much the original matrix has been refined. r_{max} is the maximal rank of all the blocks of the matrix, while b_{diag} is the size of the diagonal blocks.

Under some assumptions on how the partition $\mathfrak{S}(\mathscr{I})$ is built [19, Lemma 4.5], the sparsity constant can be bound by $O(1)$ in the $\mathscr{H}$ case. By recursively refining the diagonal blocks until they become of constant size, we have $b_{diag} = O(1)$ and $L = O(\log m)$. Therefore, (16) and (17) lead to $\mathscr{S}_{ds}^{\mathscr{H}}(m) = O(rm \log m)$ and $\mathscr{F}_{ds}^{\mathscr{H}}(m) = O(r^2 m \log^2 m)$.

In the BLR case, we have $L = 1$ and $b_{diag} = b$; the sparsity constant is equal to the number of blocks per row m/b, a high value that translates the fact that BLR matrices are much more refined than hierarchical ones. This leads to $\mathscr{S}_{ds}^{\mathscr{H},1}(m) = O(m^2)$ and $\mathscr{F}_{ds}^{\mathscr{H},1}(m) = O(m^4/b) \geq O(m^3)$, where the notation $\mathscr{S}_{ds}^{\mathscr{H},1}$ and $\mathscr{F}_{ds}^{\mathscr{H},1}$ signifies "$\mathscr{H}$ complexity formula applied to the monolevel BLR case". We thus obtain a very pessimistic complexity bound, which is due to the fact that the diagonal blocks are of the same size as all the other blocks, i.e., $b_{diag} = b$.

Applying the formulas to the MBLR case leads to the same problem. Assuming that $r = O(1)$ for the sake of simplicity, let us consider the two-level case and denote by b_1 and b_2 the outer and inner blocks sizes, respectively. We have $L = 2$, $b_{diag} = b_2$, and $c_{sp} = \max(m/b_1, b_1/b_2)$. The storage complexity

$$\mathscr{S}_{ds}^{\mathscr{H},2}(m) = O(\max(m^2 b_2/b_1, m b_1))$$

is minimized for $b_1 = O(\sqrt{m})$ and $b_2 = O(1)$, and equal to $O(m^{3/2})$, which is not a satisfying result since we have proven in Section 2.4 that this is the BLR complexity. A similarly unsatisfying result is obtained for the flop complexity, again minimized for $b_1 = O(\sqrt{m})$ and $b_2 = O(1)$ and equal to

$$\mathscr{F}_{ds}^{\mathscr{H},2}(m) = O(\max(m^4 b_2^2/b_1^3, m b_1^3/b_2)) = O(m^{5/2}).$$

Does the problem persist with a higher number of levels ℓ? The answer is unfortunately positive. Indeed, we have

$$\mathscr{S}_{ds}^{\mathscr{H},\ell}(m) = O(\max_{i=0,\ell-1}(\frac{m b_i b_\ell}{b_{i+1}})),$$

with the notation $b_0 = m$. To minimize the expression, we equilibrate each term in the maximum, which leads to the recursive relation $b_i^2 = b_{i-1} b_{i+1}$. Noting $b_1 = b$ the outer block size, it is clear that the closed-form expression of b_i is $b_i = b^i/m^{i-1}$. Since the smallest block size b_l must be at least $O(1)$, this leads to

$$\frac{b^\ell}{m^{\ell-1}} \geq O(1)$$

and thus $b \geq O(m^{(\ell-1)/\ell})$, which in turn leads to the complexity bound

$$\mathscr{S}_{ds}^{\mathscr{H},\ell}(m) \geq O(\frac{m^2 b_\ell}{b_1}) \geq O(m^{(\ell+1)/\ell}).$$

In Section 5, we will prove that for $r = O(1)$ we have $\mathscr{S}_{ds}^{\ell}(m) = O(m^{(\ell+2)/(\ell+1)})$, which is a much better bound. The analysis for the flop complexity leads to the same expression of b_i and to the complexity bound $\mathscr{F}_{ds}^{\mathscr{H},\ell}(m) \geq O(m^{(\ell+3)/\ell})$, which is again overly pessimistic compared with the bound $\mathscr{F}_{ds}^{\ell}(m) = O(m^{(\ell+3)/(\ell+1)})$ that we will prove in Section 5.

We summarize the result of applying the $\mathscr{H}$ complexity formulas in Table 4.

It is therefore clear that we must develop a new theory extending the $\mathscr{H}$ formalism to be able to prove better MBLR complexity bounds, just as we did in [4] for the BLR case. We begin by the two-level case in the next section.

4 Two-level BLR matrices

In this section, we compute the theoretical complexity of the two-level BLR factorization. The proofs and computations on this particular two-level case are meant to be illustrative of those of the general multilevel case with an arbitrary number of levels, which is discussed in Section 5.

We remind that in this section, we are considering the weakly-admissible case only.

Table 4: Applying the $\mathscr{H}$ theoretical complexity formulas to the BLR or MBLR cases does not provide a satisfying result. We have assumed that $r = O(1)$ for the sake of clarity, but the bounds would remain pessimistic for general ranks.

	$\mathscr{H}$	BLR	MBLR
L	$O(\log m)$	1	ℓ
c_{sp}	$O(1)$	m/b	$\max_{i=1,\ell} \frac{b_{i-1}}{b_i}$ *
b_{diag}	$O(1)$	b	b_ℓ
$\mathscr{S}_{ds}^{\mathscr{H}}$	$O(m \log m)$	$O(m^2)$	$O(m^{(\ell+1)/\ell})$
$\mathscr{F}_{ds}^{\mathscr{H}}$	$O(m \log^2 m)$	$O(m^3)$	$O(m^{(\ell+3)/\ell})$

*with the notation $b_0 = m$

4.1 Two-level kernels description

In order to adapt Algorithm 1 to two-level BLR matrices, two modifications must be performed.

First, the Factor kernel is not a full-rank factorization of the diagonal blocks F_{kk} anymore, but a BLR factorization. Thus, line 14 must be replaced by

$$\widetilde{F}_{kk} = \widetilde{L}_{kk}\widetilde{U}_{kk} = \text{BLR-Factor}(F_{kk}), \tag{18}$$

where BLR-Factor refers to the BLR factorization described in Algorithm 1.

Second, the FR-LR Solve kernel at lines 16 and 17 must be changed to a BLR-LR Solve:

$$\widetilde{F}_{ik} \leftarrow \text{BLR-LR-Solve}\left(\widetilde{U}_{kk}, \widetilde{F}_{ik}\right) \tag{19}$$

$$\widetilde{F}_{ki} \leftarrow \text{BLR-LR-Solve}\left(\widetilde{L}_{kk}, \widetilde{F}_{ki}\right), \tag{20}$$

where $\widetilde{F}_{ik}$ and $\widetilde{F}_{ki}$ are LR matrices and $\widetilde{L}_{kk}$ and $\widetilde{U}_{kk}$ are lower and upper triangular BLR matrices.

We describe in Algorithm 2 the BLR-LR-Solve kernel, in the upper triangular case, omitting the lower triangular case which is very similar. The kernel consists in applying a triangular solve with a upper triangular BLR matrix $\widetilde{U}$ to a low-rank matrix $\widetilde{B} = \Phi\Psi^T$. We remind that $\widetilde{U}_{ij}$ denotes the (i,j)-th low-rank sub-block of $\widetilde{U}$ (with the notation $\widetilde{U}_{jj} = U_{jj}$), and that $\Psi_{i,:}$ denotes the i-th block-row of Ψ.

Algorithm 2 BLR-LR-Solve kernel (upper triangular case)

Input: a $p \times p$ block upper triangular BLR matrix $\widetilde{U}$; $\widetilde{U} = \left[\widetilde{U}_{ij}\right]_{i=1:j,j=1:p}$ such that $\widetilde{U}_{ij} = Y_{ij}X_{ij}^T$ for $i \neq j$ and $\widetilde{U}_{jj} = U_{jj}$ is a full-rank matrix; a LR matrix $\widetilde{B} = \Phi\Psi^T$.
Output: overwritten Ψ (modified in-place) corresponding to the operation $\widetilde{B} \leftarrow \widetilde{B}\widetilde{U}^{-1}$.

1: **for** $j = 1$ **to** p **do**
2: $\quad \Psi_{j,:}^T \leftarrow \Psi_{j,:}^T - \sum_{i=1}^{j-1} \Psi_{i,:}^T Y_{ij} X_{ij}^T$
3: $\quad \Psi_{j,:}^T \leftarrow \Psi_{j,:}^T U_{jj}^{-1}$
4: **end for**

Two main operations must be performed: a triangular solve using the full-rank diagonal blocks U_{ii} of $\widetilde{U}$, and an update using the low-rank off-diagonal blocks $\widetilde{U}_{ij} = Y_{ij}X_{ij}^T$ of $\widetilde{U}$. Both are applied on the low-rank block-columns $\widetilde{B}_{:,i} = \Phi\Psi_{i,:}^T$ of $\widetilde{B}$. These two operations take place at lines 3 and 2 of Algorithm 2, respectively.

The FR-LR triangular solve can be written as

$$\widetilde{B}_{:,i} \leftarrow \widetilde{B}_{:,i}U_{ii}^{-1} = \Phi\left(\Psi_{i,:}^T U_{ii}^{-1}\right) \tag{21}$$

and thus only $\Psi_{i,:}^T$ needs to be updated, as shown at line 3.

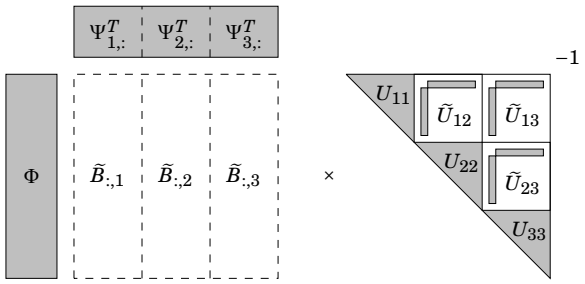


Figure 6: The BLR-LR-Solve kernel

The LR-LR update takes the following form:

$$\begin{aligned}\widetilde{B}_{:,i} \leftarrow \widetilde{B}_{:,i} - \Sigma_{j=1}^{i-1} \widetilde{B}_{:,i}\widetilde{U}_{ij} \quad &= \Phi\Psi^T_{i,:} - \Sigma_{j=1}^{i-1} \Phi\left(\Psi^T_{i,:} Y_{ij} X^T_{ij}\right) \\ &= \Phi\left(\Psi^T_{i,:} - \Sigma_{j=1}^{i-1} \Psi^T_{i,:} Y_{ij} X^T_{ij}\right)\end{aligned} \tag{22}$$

and thus, again, only $\Psi^T_{i,:}$ needs to be updated, as shown at line 2.

The BLR-LR-Solve kernel is illustrated in Figure 6.

It can easily be computed that the cost of applying the BLR-LR-Solve kernel once is equal to the storage complexity times the rank bound r:

$$cost^2_{Solve} = O(r) \times \mathscr{S}^1_{ds}(b,r). \tag{23}$$

Injecting (12) into (23) leads to

$$cost^2_{Solve} = O(b^{3/2} r^{3/2}). \tag{24}$$

4.2 Two-level complexity analysis

We now compute the complexity of the two-level BLR factorization of a dense matrix F of order m. We reuse the monolevel notations for the top level. We note b the size of the first-level blocks and $p = m/b$ the number of of block-rows and block-columns. We assume that b is of the form $b = m^x$, for $x \in [0,1]$. We also assume that the ranks of the off-diagonal blocks are bounded by $r = O(m^\alpha)$.

The size required to store the factors is again the sum of the storage for the diagonal and off-diagonal blocks, as in (8). The difference is that the diagonal blocks are not full-rank but BLR matrices. They are further refined into smaller blocks whose size should be chosen of order $O(\sqrt{br})$, as determined by (11) in the BLR complexity analysis. Therefore, in the two-level case, (8) becomes

$$\mathscr{S}^2_{ds}(b,p,r) = p \times \mathscr{S}^1_{ds}(b,r) + O(p^2 br). \tag{25}$$

By replacing $\mathscr{S}^1_{ds}(b,r)$ by its second expression computed in (12), we obtain

$$\mathscr{S}^2_{ds}(b,p,r) = O(pb^{3/2} r^{1/2}) + O(p^2 br) \tag{26}$$

Finally, we replace b, p, and r by their expression $O(m^x)$, $O(m^{1-x})$, and $O(m^\alpha)$, respectively, to obtain

$$\mathscr{S}^2_{ds}(m,x,\alpha) = O(m^{1+(x+\alpha)/2} + m^{2-x+\alpha}). \tag{27}$$

This leads to

$$x^* = (2+\alpha)/3, \tag{28}$$

where x^* defines the optimal choice of the first-level block size b. We thus obtain a final two-level storage complexity of

$$\mathscr{S}^2_{ds}(m,r) = O(m^{4/3} r^{2/3}). \tag{29}$$

Table 5: Two-level equivalent of Table 3. The legend of the table is the same. The differences between the two tables are highlighted in gray.

Step	Type	$cost^2_{Step}$	$number_{Step}$	$\mathscr{F}^2_{Step}(b,p,r)$	=	$\mathscr{F}^2_{Step}(m,x,\alpha)$
Factor	BLR	$O(b^2r)$	$O(p)$	$O(pb^2r)$	=	$O(m^{1+x+\alpha})$
Solve	BLR-LR	$O(b^{3/2}r^{3/2})$	$O(p^2)$	$O(p^2b^{3/2}r^{3/2})$	=	$O(m^{2-x/2+3\alpha/2})$
Product	LR-LR	$O(br^2)$	$O(p^3)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$
Recompress	LR	$O(bpr^2)$	$O(p^2)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$

We now compute the flop complexity $\mathscr{F}^2_{ds}(m,r)$ of the two-level BLR dense factorization. We compute the cost of each step and report it in Table 5, which is the two-level equivalent of Table 3; the differences between the two tables are highlighted in gray. The Product and Recompress steps have not changed and thus have the same cost. The Factor step consists in factorizing the p diagonal blocks which are now represented by BLR matrices; thus its cost is directly derived from the BLR complexity computed in (15):

$$\mathscr{F}^2_{Factor}(b,p,r) = p \times \mathscr{F}^1_{ds}(b,r) = O(pb^2r). \tag{30}$$

It remains to compute the cost of the Solve step, which now takes the form of $O(p^2)$ calls to the BLR-LR-Solve kernel whose cost is given by (24). Thus, the overall cost of the Solve step is

$$\mathscr{F}^2_{Solve}(b,p,r) = O(p^2b^{3/2}r^{3/2}). \tag{31}$$

This concludes the computations for the third, fourth, and fifth columns of Table 5. Just as in the monolevel case, the sixth column is obtained by replacing b, p, and r by their expression $O(m^x)$, $O(m^{1-x})$, and $O(m^\alpha)$, respectively.

We can finally compute the total flop complexity as the sum of the costs of all steps

$$\mathscr{F}^2_{ds}(m,x,\alpha) = O(m^{1+x+\alpha} + m^{2-x/2+3\alpha/2} + m^{3-2x+2\alpha}) \tag{32}$$

We then compute x^* which is again equal to the same value as the one that minimizes the storage complexity, $x^* = (2+\alpha)/3$. Therefore, the final two-level dense flop complexity is

$$\mathscr{F}^2_{ds}(m,r) = O(m^{5/3}r^{4/3}). \tag{33}$$

The two-level BLR format therefore significantly improves the asymptotic storage and flop complexity compared with the monolevel format. Our analysis shows that the top level block size should be set to $b^* = O(m^{x^*}) = O(m^{2/3}r^{1/3})$. This is an asymptotically larger value than the monolevel optimal block size computed in (11), which translates the fact that by refining the diagonal blocks, we can afford to take a larger block size to improve the overall asymptotic complexity. However, contrarily to hierarchical matrices, b^* remains asymptotically much lower than $O(m)$; this makes the format much more flexible for the reasons described in Section 2.5. In short, *we have traded off some of the flexibility of the monolevel format to improve the asymptotic complexity.*

This improvement of the dense flop and storage complexities is translated into an improvement of the sparse complexities. Assuming a rank bound in $O(1)$, we quantify this improvement in Table 6. Compared with the monolevel BLR format, the two-level BLR format drops the $O(\log n)$ factor in the 2D flop and 3D storage complexities, which become linear and thus optimal. The two-level BLR format can thus achieve, in these two cases, the same $O(n)$ complexity as the hierarchical formats while being almost as simple and flexible as flat formats.

Finally, the 3D flop complexity remains superlinear but is significantly reduced, from $O(n^{4/3})$ to $O(n^{10/9})$. As the problem size gets larger and larger, even small asymptotic improvements can make a big difference. Therefore, we now generalize the two-level analysis to the multilevel case with an arbitrary number of levels.

Table 6: Flop and storage complexities of the monolevel and two-level BLR factorizations of a sparse system of $n = N \times N$ (2D case) or $n = N \times N \times N$ (3D case) unknowns, derived from the complexities of the factorization of a dense matrix of order m. We consider a constant rank bound $r = O(1)$. The BLR variant considered is CUFS.

	$\mathscr{F}_{ds}(m)$	$\mathscr{F}_{sp}(n)$		$\mathscr{S}_{ds}(m)$	$\mathscr{S}_{sp}(n)$	
		2D	3D		2D	3D
Monolevel BLR	$O(m^2)$	$O(n \log n)$	$O(n^{4/3})$	$O(m^{1.5})$	$O(n)$	$O(n \log n)$
Two-level BLR	$O(m^{5/3})$	$O(n)$	$O(n^{10/9})$	$O(m^{4/3})$	$O(n)$	$O(n)$

5 Generalization to ℓ-level BLR matrices

In this section, we generalize two-level proof and computations of the previous section to an arbitrary number of levels ℓ by computing recursive complexity formulas. We remind that in this section, we are still considering the weakly-admissible case only.

5.1 Recursive complexity analysis

Just as for the two-level case, one can compute the three-level asymptotic complexities, and so on until the general formula becomes clear. We state the result for an arbitrary number of levels ℓ in the following theorem.

Theorem 1 (Storage and flop complexity of the ℓ-level BLR factorization)**.** *Let us consider a dense ℓ-level BLR matrix of order m. We note b the size of the top level blocks, and $p = m/b$. Let $r = O(m^\alpha)$ be the bound on the maximal rank of any block on any level. Then, the optimal choice of the top level block size is $b = O(m^{x^*})$, with $x^* = (\ell + \alpha)/(\ell + 1)$, which leads to the following storage and flop complexities:*

$$\mathscr{S}^\ell_{ds}(m,r) = O(m^{(\ell+2)/(\ell+1)} r^{\ell/(\ell+1)}); \tag{34}$$

$$\mathscr{F}^\ell_{ds}(m,r) = O(m^{(\ell+3)/(\ell+1)} r^{2\ell/(\ell+1)}). \tag{35}$$

Proof. The proof is inductive. The formulas hold for $\ell = 1$ and 2, as proven in the Sections 2.4 and 4.2, respectively. Let us assume that they are true for the $(\ell-1)$-level BLR factorization and prove that they still hold for the ℓ-level one.

$\mathscr{S}^\ell_{ds}(m,r)$ can be computed as the storage cost for the $O(p^2)$ off-diagonal low-rank blocks plus that of the $O(p)$ diagonal blocks, which are represented as $(\ell-1)$-level BLR matrices of order b and with blocks of rank at most $r = O(m^\alpha)$:

$$\mathscr{S}^\ell_{ds}(b,p,r) = p \times \mathscr{S}^{\ell-1}_{ds}(b,r) + O(p^2 br). \tag{36}$$

By induction, the previous equation becomes

$$\mathscr{S}^\ell_{ds}(b,p,r) = p \times O(b^{(\ell+1)/\ell} r^{(\ell-1)/\ell}) + O(p^2 br). \tag{37}$$

We replace b, p, and r by their expression $O(m^x)$, $O(m^{1-x})$, and $O(m^\alpha)$, respectively, to obtain

$$\mathscr{S}^\ell_{ds}(m,x,\alpha) = O(m^{1-x+(\ell+1)x/\ell+\alpha(\ell-1)/\ell}) + O(m^{2-x+\alpha}). \tag{38}$$

$\mathscr{S}^\ell_{ds}(m,x,\alpha)$ is therefore minimized for an x^* satisfying

$$1 - x^* + (\ell+1)x^*/\ell + \alpha(\ell-1)/\ell = 2 - x^* + \alpha, \tag{39}$$

which leads to

$$x^* = (\ell + \alpha)/(\ell + 1) \tag{40}$$

Table 7: ℓ-level equivalent of Tables 3 and 5. The legend of the table is the same. The differences between this table and the previous two are highlighted in gray.

Step	Type	$cost^{\ell}_{Step}$	$number_{Step}$	$\mathscr{F}^{\ell}_{Step}(b,p,r)$	=	$\mathscr{F}^{\ell}_{Step}(m,x,\alpha)$
Factor	$\mathrm{BLR}_{(\ell-1)}$	$O(b^{(\ell+2)/\ell}r^{2(\ell-1)/\ell})$	$O(p)$	$O(pb^{(\ell+2)/\ell}r^{2(\ell-1)/\ell})$	=	$O(m^{1+2x/\ell+2\alpha(\ell-1)/\ell})$
Solve	$\mathrm{BLR}_{(\ell-1)}$-LR	$O(b^{(\ell+1)/\ell}r^{(2\ell-1)/\ell})$	$O(p^2)$	$O(p^2b^{(\ell+1)/\ell}r^{(2\ell-1)/\ell})$	=	$O(m^{2+(1-\ell)x/\ell+\alpha(2\ell-1)/\ell})$
Product	LR-LR	$O(br^2)$	$O(p^3)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$
Recompress	LR	$O(bpr^2)$	$O(p^2)$	$O(p^3br^2)$	=	$O(m^{3-2x+2\alpha})$

and thus (34) stands proven.

Similarly to the two-level case, $\mathscr{F}^{\ell}_{ds}(m,r)$ can be computed as the sum of the costs of the Factor, Solve, Product and Recompress steps. We provide the ℓ-level equivalent of Tables 3 and 5 in Table 7. As previously, the costs of the Product and Recompress steps do not depend on ℓ and are simply equal to $O(p^3br^2)$, and thus

$$\mathscr{F}^{\ell}_{Product}(m,\alpha,x) = \mathscr{F}^{\ell}_{Recompress}(m,\alpha,x) = O(m^{3-2x+2\alpha}). \tag{41}$$

The Solve step consists in applying $O(p^2)$ times the $\mathrm{BLR}_{(\ell-1)}$-LR-Solve kernel, described in Algorithm 3, where $\mathrm{BLR}_{(\ell-1)}$ denotes a $(\ell-1)$-level BLR matrix (with the convention that BLR_0 denotes a FR matrix). It can easily be proven by induction that the cost of applying the $\mathrm{BLR}_{(\ell-1)}$-LR-Solve kernel is $O(r)\times\mathscr{S}^{\ell-1}_{ds}(b,r)$, just as in the two-level case (see (23)). Therefore, it holds

$$\mathscr{F}^{\ell}_{Solve}(p,b,r) = O(p^2r)\times\mathscr{S}^{\ell-1}_{ds}(b,r), \tag{42}$$

which, according to (34) which we have already proven, leads to

$$\mathscr{F}^{\ell}_{Solve}(p,b,r) = O(p^2b^{(\ell+1)/\ell}r^{(2\ell-1)/\ell}) \tag{43}$$

and thus

$$\mathscr{F}^{\ell}_{Solve}(m,\alpha,x) = O(m^{2+(1-\ell)x/\ell+\alpha(2\ell-1)/\ell}). \tag{44}$$

The Factor step consists in factorizing p diagonal $(\ell-1)$-level BLR matrices of order b and rank at most $r = O(m^{\alpha})$. Therefore,

$$\mathscr{F}^{\ell}_{Factor}(p,b,r) = p\times\mathscr{F}^{\ell-1}_{ds}(b,r), \tag{45}$$

which, by induction, leads to

$$\mathscr{F}^{\ell}_{Factor}(p,b,r) = O(pb^{(\ell+2)/\ell}r^{2(\ell-1)/\ell}) \tag{46}$$

and thus

$$\mathscr{F}^{\ell}_{Factor}(m,\alpha,x) = O(m^{1+2x/\ell+2\alpha(\ell-1)/\ell}). \tag{47}$$

Finally, we compute x^* as the value of x that minimizes the sum of the cost of all steps (given by (41), (44), and (47)). We find again

$$x^* = (\ell+\alpha)/(\ell+1) \tag{48}$$

and therefore (35) stands proven. □

5.2 Influence of the number of levels ℓ

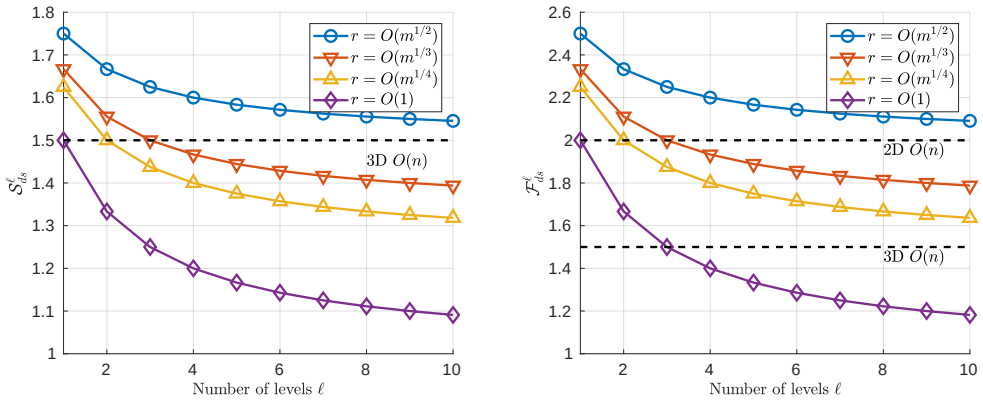
With the formulas from Theorem 1 proven, we now analyze their practical implications. It is clear that both the storage and flop asymptotic complexities decrease monotonically with the number of levels, while the top level block size increases.

In Figure 7, we plot the value of the exponent of the asymptotic complexities as a function of the number of levels, for different rank bounds $r = O(m^{\alpha})$.

Algorithm 3 BLR_ℓ-LR-Solve kernel, $\ell > 1$ (upper triangular case)

Input: a $p \times p$ block upper triangular BLR_ℓ matrix $\widetilde{U}$; $\widetilde{U} = \left[\widetilde{U}_{ij}\right]_{i=1:j,j=1:p}$ such that $\widetilde{U}_{ij} = Y_{ij}X_{ij}^T$ for $i \neq j$ and $\widetilde{U}_{jj}$ is a $\text{BLR}_{(\ell-1)}$ matrix; a LR matrix $\widetilde{B} = \Phi\Psi^T$.
Output: overwritten Ψ (modified in-place) corresponding to the operation $\widetilde{B} \leftarrow \widetilde{B}\widetilde{U}^{-1}$.

1: **for** $j = 1$ **to** p **do**
2: $\quad \Psi_{j,:}^T \leftarrow \Psi_{j,:}^T - \sum_{i=1}^{j-1} \Psi_{i,:}^T Y_{ij} X_{ij}^T$
3: $\quad \text{BLR}_{(\ell-1)}\text{-LR-Solve}\left(\widetilde{U}_{jj}, \Phi\Psi_{j,:}^T\right)$
4: **end for**



(a) Storage complexity. (b) Flop complexity.

Figure 7: Theoretical asymptotic exponent of the storage and flop complexity of the dense MBLR factorization

Let us first consider the case of $r = O(1)$ (i.e., $\alpha = 0$). We have already shown that, in this case, the two-level BLR factorization has $O(m^{5/3})$ flop complexity, which leads to $O(n^{10/9})$ 3D sparse complexity. With a third level, the dense complexity decreases to $O(m^{3/2})$, which is precisely the 3D sparse breaking point (see Table 1), and thus leads to $O(n \log n)$ complexity. The log factor can be dropped by adding a fourth level. The dense complexity tends towards $O(m)$ as the number of levels increases, but the sparse complexity cannot be further improved after the optimal $O(n)$ has been reached. This illustrates that only a small number of levels is necessary to reach low sparse complexity. In particular, with four levels, the number of blocks on the top level is $p = O(m^{1/5})$, which is still a quite large number which, as previously described, provides more flexibility to address issues such as data distribution, parallel implementation, numerical pivoting, etc.

The picture is different with higher rank bounds. Indeed, the higher the rank, the more difficult it is to reach a low complexity and thus more levels are required. For example, with $r = O(\sqrt{m})$ ($\alpha = 1/2$), it is actually not possible to reach a $O(n)$ 3D flop complexity since the dense complexity is at best $O(mr^2) = O(m^2)$. This is the dense complexity achieved by the hierarchical formats, as well as the MBLR format with an infinite number of levels, and leads to $O(n^{4/3})$ sparse complexity. It is therefore not possible to achieve this complexity with a constant number of levels. However, a crucial observation is that the rate of improvement of the exponent, which follows $(\ell + 3 + 2\alpha\ell)/(\ell + 1)$, is rapidly decreasing as ℓ increases. For example, with $\alpha = 1/2$, one level decreases the full-rank $O(m^3)$ complexity to the BLR $O(m^{2.5})$ complexity; it would require an infinite number of levels to achieve another $O(m^{0.5})$ factor of gain. Similarly, adding two more levels leads to $O(m^{2.25})$, achieving a $O(m^{0.25})$ gain which can only be achieved again with infinitely more levels! This illustrates the critical observation that *the first few levels achieve most of the asymptotic gain*. We therefore believe that the MBLR factorization with only a small number of levels can be of practical interest, even for problems with larger ranks.

6 Numerical experiments

In this section we compare the experimental complexities of the full-rank, BLR, and MBLR formats (with different numbers of levels) for the factorization of dense matrices arising from Schur complements of sparse problems.

6.1 Experimental setting

We have developed a MATLAB code to perform the BLR and MBLR factorization of a dense matrix, which we use to run all experiments. Numerical experiments with sparse factorizations are out of the scope of this article.

To compute the low-rank form of the blocks, we perform a truncated QR factorization with column pivoting (i.e., a truncated version of LAPACK's [8] `_geqp3` routine). We use a a mix of fixed-accuracy and fixed-rank truncation: we stop the factorization after either an accuracy of ε has been achieved or at most r_{max} columns have been computed. In the following experiments, we have set $\varepsilon = 10^{-14}$ and $r_{max} = 10$.

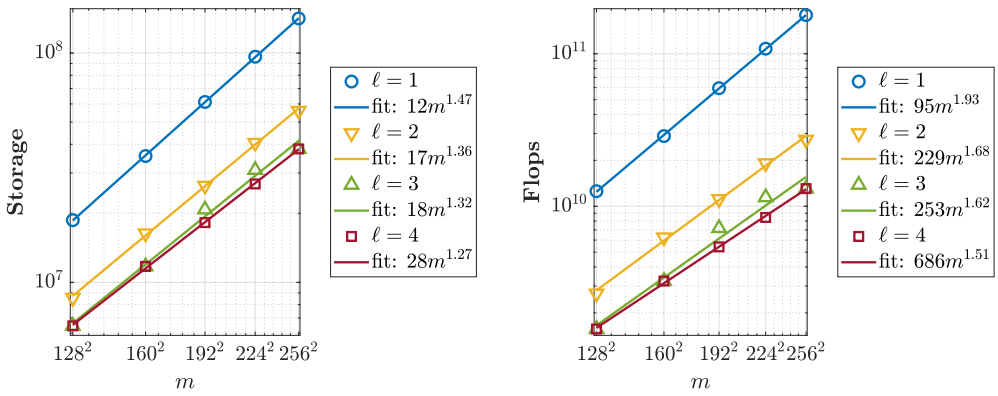
All the experiments were performed on the brunch computer from the LIP laboratory (ENS Lyon), a shared-memory machine equipped with 24 Haswell processors and 1.5 TB of memory.

To validate our theoretical complexity results, we use a Poisson problem, which generates the symmetric positive definite matrix A from a 7-point finite-difference discretization of equation

$$\Delta u = f$$

on a 3D domain of size $n = N \times N \times N$ with Dirichlet boundary conditions. We compute the dense MBLR factorization of the matrices corresponding to the root separator of the nested dissection partitioning, which are of order $m = N^2$.

We compute the experimental asymptotic complexities by means of the least-squares estimation of the coefficients $\{\beta_i\}_i$ of a regression function f such that $X_{fit} = f(N, \{\beta_i\}_i)$ fits the observed data



(a) Storage complexity.

(b) Flop complexity.

Figure 8: Experimental complexity of the dense MBLR factorization of the root separator of order $m = N^2$ of a 3D Poisson problem of order $n = N^3$.

Table 8: Comparison between theoretical and experimental complexities.

	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
		Storage complexity		
Theoretical	$O(m^{1.50})$	$O(m^{1.33})$	$O(m^{1.25})$	$O(m^{1.20})$
Experimental	$O(m^{1.47})$	$O(m^{1.36})$	$O(m^{1.32})$	$O(m^{1.27})$
		Flop complexity		
Theoretical	$O(m^{2.00})$	$O(m^{1.67})$	$O(m^{1.50})$	$O(m^{1.40})$
Experimental	$O(m^{1.97})$	$O(m^{1.68})$	$O(m^{1.62})$	$O(m^{1.51})$

X_{obs}. We use the following regression function:

$$X_{fit} = e^{\beta_1^*} N^{\beta_2^*} \text{ with } \beta_1^*, \beta_2^* = \underset{\beta_1,\beta_2}{\arg\min} \| \log X_{obs} - \beta_1 - \beta_2 \log N \|^2.$$

6.2 Experimental complexity results

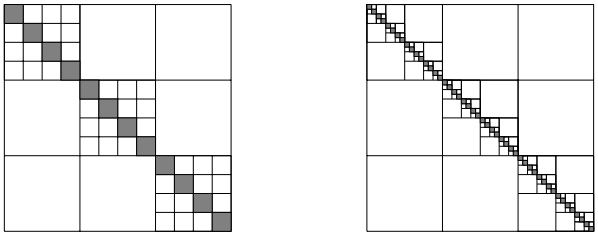
We report in Figure 8 the experimental storage and flop complexity of the dense MBLR factorization, using a number of levels ℓ varying from 1 to 4. We summarize the asymptotic exponents obtained by fitting the data points in Table 8. The experimental complexities are in relatively good agreement with the theoretical ones.

While there is an almost perfect match for the first two levels, the gap between theory and practice increases as more levels are added. This is due to the increasing difficulty of tuning the different block sizes at each level. This is a practical limitation that should be further investigated. Nevertheless, an asymptotic gain is achieved by each addition of a new level, at least up to $\ell = 4$.

Overall, these experimental results therefore support the capacity of the MBLR format to significantly reduce the asymptotic complexity of the factorization, even when only a small number of levels is used.

7 Comparison with the BLR-$\mathscr{H}$ format

In this section, we compare our MBLR format to the related BLR-$\mathscr{H}$ format, sometimes also referred to as "Lattice-$\mathscr{H}$". It consists in representing the matrix using the BLR format, and then



(a) MBLR format. (b) BLR-$\mathscr{H}$ format.

Figure 9: Comparison of the MBLR and BLR-$\mathscr{H}$ formats.

approximating its diagonal blocks with $\mathscr{H}$-matrices. This is illustrated in Figure 9: contrarily to our MBLR format (Figure 9(a)), the BLR-$\mathscr{H}$ format (Figure 9(b)) uses the more complex $\mathscr{H}$ representation to approximate the diagonal blocks.

The BLR-$\mathscr{H}$ format has been little studied from a theoretical standpoint in the literature, even though it has been considered as a simple way to use hierarchical matrices in a distributed-memory setting [22, 1].

Since the MBLR and BLR-$\mathscr{H}$ formats have essentially the same objective, in this section we provide a theoretical analysis comparing them in terms of asymptotic complexity and simplicity of the format.

As illustrated in Figure 9, the BLR-$\mathscr{H}$ format is obviously more complex than the MBLR one, since the diagonal blocks are represented as $\mathscr{H}$-matrices and thus refined with a nonconstant number of levels, rather than a constant one. The question is thus whether these additional levels improve the asymptotic complexity of the format. In the following, we prove it is not the case and therefore recommend the use of the MBLR format over that of the BLR-$\mathscr{H}$ one.

The complexity of the BLR-$\mathscr{H}$ format is entirely determined by the block size b used for the BLR partitioning. Indeed, the storage complexity can be computed as the sum of the storage for the off-diagonal low-rank blocks and that of the diagonal $\mathscr{H}$ blocks:

$$\mathscr{S}_{ds}^{BLR-\mathscr{H}}(p,b,r) = O(p^2br) + O(pbr\log b) = O(p^2br),$$

where $p = m/b$ and q is a small integer. Thus, the term corresponding to the off-diagonal low-rank blocks is dominant and we obtain

$$\mathscr{S}_{ds}^{BLR-\mathscr{H}}(m,b,r) = O(\frac{m^2r}{b}). \tag{49}$$

Similarly, the flop complexity of the BLR-$\mathscr{H}$ factorization can be computed as the sum of the cost of the $\mathscr{H}$-Factor, $\mathscr{H}$-LR-Solve, LR-LR-Product, and Recompress steps:

$$\mathscr{F}_{ds}^{BLR-\mathscr{H}}(p,b,r) = O(pbr^2\log^2 b) + O(p^2br^2\log b) + O(p^3br^2) + O(p^3br^2) = O(p^3br^2),$$

which is dominated by the LR-LR-Product and Recompress steps and therefore leads to

$$\mathscr{F}_{ds}^{BLR-\mathscr{H}}(m,b,r) = O(\frac{m^3r^2}{b^2}). \tag{50}$$

The question is now how this result compares to the MBLR complexities computed in the previous sections. Let us first take an example: assume that the matrix has been partitioned in blocks of size $b = \sqrt{mr}$. Then, applying (49) and (50), we obtain the same asymptotic complexity as that of the monolevel BLR format:

$$\mathscr{S}_{ds}^{BLR-\mathscr{H}}(m,r) = \mathscr{S}_{ds}^{1}(m,r) = O(m^{3/2}r^{1/2});$$
$$\mathscr{F}_{ds}^{BLR-\mathscr{H}}(m,r) = \mathscr{F}_{ds}^{1}(m,r) = O(m^2r).$$

This result means that representing the diagonal blocks as $\mathscr{H}$ matrices does not improve the asymptotic complexity, since the BLR format (in which the diagonal blocks are stored in full-rank) already achieves the same complexity. This result generalizes independently of the choice of the block size b. Indeed, if we apply (49) and (50), with the optimal choice of block size for the ℓ-level BLR format, $b = O(m^{\ell/(\ell+1)}r^{1/(\ell+1)})$, we obtain the same asymptotic complexity as that proved in Section 5 (Theorem 1):

$$\mathscr{S}_{ds}^{BLR-\mathscr{H}}(m,r) = \mathscr{S}_{ds}^{\ell}(m,r) = O(m^{(\ell+2)/(\ell+1)}r^{\ell/(\ell+1)});$$
$$\mathscr{F}_{ds}^{BLR-\mathscr{H}}(m,r) = \mathscr{F}_{ds}^{\ell}(m,r) = O(m^{(\ell+3)/(\ell+1)}r^{2\ell/(\ell+1)}).$$

This result can be interpreted as follows: for any given block size b, there exists a constant number of levels ℓ_b such that representing the diagonal blocks of the matrix as ℓ_b-level BLR matrices suffices to achieve the lowest possible complexity. As far as asymptotic complexity is concerned, it is thus not necessary to represent these diagonal blocks with the $\mathscr{H}$ format. Note that the value of ℓ_b can easily be computed as

$$\ell_b = \min\left\{\ell : m^{\ell/(\ell+1)}r^{1/(\ell+1)} \geq b\right\} - 1.$$

Also note that if $b = O(m)$, then ℓ_b asymptotically tends towards infinity and then the MBLR and BLR-$\mathscr{H}$ formats are equivalent, in the sense that they both lead to $b = m/c$ for some $c = O(1)$. They are however still different from the typical $\mathscr{H}$ format with $c = 2$; moderately large, albeit constant, values of c (e.g., 10 or 100) could be of high practical interest.

8 Conclusion

We have proposed a new multilevel BLR (MBLR) format to bridge the gap between flat and hierarchical low-rank matrix formats. Contrarily to hierarchical formats for which the number of levels in the block hierarchy is logarithmically dependent on the size of the problem, the MBLR format only uses a constant number of levels ℓ.

We had previously explained why the $\mathscr{H}$-matrix theory, while applicable to the BLR format, leads to very pessimistic complexity bounds and is therefore not suitable. Here, we have shown that this remains true for the MBLR format and we therefore extended the theory to compute better bounds. We proved that both the storage and flop complexities of the factorization can be finely controlled by ℓ. We theoretically showed that the first few levels achieve most of the asymptotic gain that can be expected. In particular, for a sparse 3D problem with constant ranks, two levels suffice to achieve $O(n)$ storage complexity and three levels achieve $O(n \log n)$ flop complexity, suggesting that a small number of levels may be enough in practice. Our numerical experiments confirm this trend.

Having a small number of levels leads to a greater freedom to distribute data in parallel; in particular blocks are small enough for several of them to fit in shared-memory, allowing an efficient parallelization. Finally, a small number of levels greatly simplifies the implementation of the format, making it easy to handle important features such as dynamic data structures and numerical pivoting. The related BLR-$\mathscr{H}$ (or Lattice-$\mathscr{H}$) format targets a similar objective; however, our theoretical analysis shows that using more levels to refine the diagonal blocks actually does not improve the asymptotic complexity with respect to the MBLR format.

In short, the MBLR format aims to strike a balance between asymptotic complexity and actual performance on parallel computers by trading off the optimal hierarchical complexity to retrieve some of the simplicity and flexibility of flat monolevel formats. We believe that this increased simplicity and flexibility will prove to be useful in a parallel, algebraic, fully-featured, general purpose sparse direct solver. The implementation of the MBLR format in such a solver will be the object of future work.

Acknowledgements

We thank Cleve Ashcraft for useful discussions.

References

[1] Private communication with R. Kriemann.

[2] P. R. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. Improving multifrontal methods by means of block low-rank representations. *SIAM Journal on Scientific Computing*, 37(3):A1451–A1474, 2015.

[3] P. R. Amestoy, R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, A. Miniussi, and S. Operto. Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea. *Geophysics*, 81(6):R363 – R383, 2016.

[4] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. On the complexity of the Block Low-Rank multifrontal factorization. *SIAM Journal on Scientific Computing*, 39(4):A1710–A1740, 2017.

[5] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. 2017. submitted to *ACM Transactions on Mathematical Software*.

[6] A. Aminfar, S. Ambikasaran, and E. Darve. A fast block low-rank dense solver with applications to finite-element matrices. *Journal of Computational Physics*, 304:170–188, 2016.

[7] A. Aminfar and E. Darve. A fast, memory efficient and robust sparse preconditioner based on a multifrontal approach with applications to finite-element matrices. *International Journal for Numerical Methods in Engineering*, 107(6):520–540, 2016.

[8] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM Press, Philadelphia, PA, third edition, 1995.

[9] C. Ashcraft, R. G. Grimes, J. G. Lewis, B. W. Peyton, and H. D. Simon. Progress in sparse matrix methods for large linear systems on vector computers. *Int. Journal of Supercomputer Applications*, 1(4):10–30, 1987.

[10] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer-Verlag, 2008.

[11] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Engineering analysis with boundary elements*, 27(5):405–422, 2003.

[12] S. Chandrasekaran, M. Gu, and T. Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, 28(3):603–622, 2006.

[13] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.

[14] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Transactions on Mathematical Software*, 9:302–325, 1983.

[15] J. A. George. Nested dissection of a regular finite-element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[16] P. Ghysels, X. S. Li, C. Gorman, and F. H. Rouet. A robust parallel preconditioner for indefinite systems using hierarchical matrices and randomized sampling. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 897–906, May 2017.

[17] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov. An efficient multicore implementation of a novel hss-structured multifrontal solver using randomized sampling. *SIAM Journal on Scientific Computing*, 38(5):S358–S384, 2016.

[18] A. Gillman, P. Young, and P.-G. Martinsson. A direct solver with $\mathcal{O}(N)$ complexity for integral equations on one-dimensional domains. *Frontiers of Mathematics in China*, 7:217–247, 2012.

[19] L. Grasedyck and W. Hackbusch. Construction and Arithmetics of $\mathscr{H}$-Matrices. *Computing*, 70(4):295–334, 2003.

[20] W. Hackbusch. A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: introduction to $\mathscr{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[21] W. Hackbusch. *Hierarchical matrices : algorithms and analysis*, volume 49 of *Springer series in computational mathematics*. Springer, Berlin, 2015.

[22] A. Ida. Efficient Low-rank Solver for Integral Equations on Distributed Memory Systems. In *SIAM Conference on Parallel Processing (SIAM PP18)*, Tokyo, Japan, March 2018.

[23] J. W. H. Liu. The multifrontal method for sparse matrix solution: Theory and Practice. *SIAM Review*, 34:82–109, 1992.

[24] T. Mary. *Block Low-Rank multifrontal solvers: complexity, performance, and scalability*. PhD thesis, Université de Toulouse, November 2017.

[25] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman. Sparse Supernodal Solver Using Block Low-Rank Compression. In *18th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2017)*, Orlando, United States, jun 2017.

[26] H. Pouransari, P. Coulier, and E. Darve. Fast hierarchical solvers for sparse matrices using low-rank approximation. *arXiv preprint arXiv:1510.07363*, 2015.

[27] D. Shantsev, P. Jaysaval, S. de la Kethulle de Ryhove, P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. Large-scale 3D EM modeling with a Block Low-Rank multifrontal direct solver. *Geophysical Journal International*, 209(3):1558–1571, 2017.

[28] J. Xia. Efficient structured multifrontal factorization for general large sparse matrices. *SIAM Journal on Scientific Computing*, 35(2):A832–A860, 2013.

[29] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2009.

[30] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

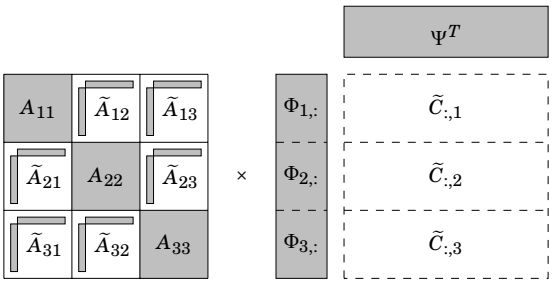


Figure 10: The BLR-LR-Product kernel.

A Appendix: extension to the strongly-admissible case

In Sections 4 and 5, we were considering the weakly-admissible case only. In this section, we extend the proofs and computations to the strongly-admissible case. This requires the introduction of new kernels involving computations on off-diagonal non-admissible blocks that are represented as MBLR matrices. We prove that the theoretical complexity results established in the weakly-admissible case still hold in the strongly-admissible one. The proof involves two main ingredients. First, the fact that the number of non-admissible blocks per row or column on any level of the block hierarchy can be bounded by a constant. Second, the computation of the cost of the new kernels, which we show do not asymptotically increase the overall cost of the factorization.

A.1 New kernels description

Let us consider an ℓ-level BLR matrix. In the strongly admissible case, not all off-diagonal blocks are admissible and thus low-rank; some must therefore be further refined by representing them by $(\ell-1)$-level BLR matrices.

This introduces three new kernels: the $\text{BLR}_{(\ell-1)}$-LR and $\text{BLR}_{(\ell-1)}$-$\text{BLR}_{(\ell-1)}$ products, and the $\text{BLR}_{(\ell-1)}$-$\text{BLR}_{(\ell-1)}$ triangular solve.

Algorithm 4 BLR_ℓ-LR-Product kernel

Input: a $p \times p$ BLR_ℓ matrix $\widetilde{A}$ and a LR matrix $\widetilde{C} = \Phi\Psi^T$.
Output: a LR matrix $\widetilde{C}^{out} = \Phi^{out}\Psi^T = \widetilde{A}\widetilde{C}$

1: **for** $i = 1$ **to** p **do**
2: $\quad \Phi^{out}_{i,:}\Psi^T \leftarrow \text{BLR}_{(\ell-1)}\text{-LR-Product}\left(\widetilde{A}_{ii}, \Phi_{i,:}\Psi^T\right)$
3: $\quad \Phi^{out}_{i,:} \leftarrow \Phi^{out}_{i,:} + \sum_{j=1;j\neq i}^{p} X_{ij}Y_{ij}^T\Phi_{j,:}$
4: **end for**

The BLR_ℓ-LR-Product is described in Algorithm 4 and illustrated (with $\ell = 1$) in Figure 10. Note that the algorithm can easily be adapted to define the similar LR-BLR_ℓ-Product. The BLR_ℓ-BLR_ℓ-Product is described in Algorithm 5. Finally, the BLR_ℓ-BLR_ℓ-Solve is described in Algorithm 6 in its upper triangular version; the lower triangular version is similar and is omitted for the sake of conciseness.

To prove that the strong admissibility case does not increase the asymptotic complexity of the ℓ-level BLR factorization, we thus need to compute the cost of these three new steps. To do so, two ingredients are necessary. First, we prove in Section A.2 that the number of non-admissible blocks at any level of the block hierarchy is bounded by a constant (Lemma 1). Then, in Section A.3, we compute their cost by induction (Lemma 2).

Algorithm 5 BLR_ℓ-BLR_ℓ-Product kernel

Input: two $p \times p$ BLR_ℓ matrices $\widetilde{A}$ and $\widetilde{B}$.
Output: a $p \times p$ BLR_ℓ matrix $\widetilde{C} = \widetilde{A}\widetilde{B}$.

for $i = 1$ **to** p **do**
  for $j = 1$ **to** p **do**
    $\widetilde{C}_{ij} \leftarrow [\,]$ (rank-0 matrix)
    for $k = 1$ **to** p **do**
      if $\widetilde{A}_{ik}$ and $\widetilde{B}_{kj}$ are LR **then**
        $\widetilde{C}_{ij} \leftarrow \widetilde{C}_{ij} + \widetilde{A}_{ik}\widetilde{B}_{kj}$
      else if $\widetilde{A}_{ik}$ is LR **then**
        $\widetilde{C}_{ij} \leftarrow \widetilde{C}_{ij} + \text{LR-BLR}_{(\ell-1)}\text{-Product}\left(\widetilde{A}_{ik}, \widetilde{B}_{kj}\right)$
      else if $\widetilde{B}_{kj}$ is LR **then**
        $\widetilde{C}_{ij} \leftarrow \widetilde{C}_{ij} + \text{BLR}_{(\ell-1)}\text{-LR-Product}\left(\widetilde{A}_{ik}, \widetilde{B}_{kj}\right)$
      else
        $\widetilde{C}_{ij} \leftarrow \widetilde{C}_{ij} + \text{BLR}_{(\ell-1)}\text{-BLR}_{(\ell-1)}\text{-Product}\left(\widetilde{A}_{ik}, \widetilde{B}_{kj}\right)$
      end if
    end for
  end for
end for

Algorithm 6 BLR_ℓ-BLR_ℓ-Solve kernel (upper triangular case)

Input: two $p \times p$ BLR_ℓ matrices $\widetilde{U}$ and $\widetilde{A}$; $\widetilde{U}$ is upper triangular.
Output: overwritten $\widetilde{A}$ (modified in-place) corresponding to the operation $\widetilde{A} \leftarrow \widetilde{A}\widetilde{U}^{-1}$.

for $k = 1$ **to** p **do**
  for $i = 1$ **to** p **do**
    if $\widetilde{A}_{ik}$ is LR **then**
      $\widetilde{A}_{ik} \leftarrow \text{BLR}_{(\ell-1)}\text{-LR-Solve}\left(\widetilde{U}_{kk}, \widetilde{A}_{ik}\right)$
    else
      $\widetilde{A}_{ik} \leftarrow \text{BLR}_{(\ell-1)}\text{-BLR}_{(\ell-1)}\text{-Solve}\left(\widetilde{U}_{kk}, \widetilde{A}_{ik}\right)$
    end if
    for $j = k + 1$ **to** p **do**
      if $\widetilde{A}_{ik}$ and $\widetilde{U}_{kj}$ are LR **then**
        $\widetilde{A}_{ij} \leftarrow \widetilde{A}_{ij} - \widetilde{A}_{ik}\widetilde{U}_{kj}$
      else if $\widetilde{A}_{ik}$ is LR **then**
        $\widetilde{A}_{ij} \leftarrow \widetilde{A}_{ij} - \text{LR-BLR}_{(\ell-1)}\text{-Product}\left(\widetilde{A}_{ik}, \widetilde{U}_{kj}\right)$
      else if $\widetilde{U}_{kj}$ is LR **then**
        $\widetilde{A}_{ij} \leftarrow \widetilde{A}_{ij} - \text{BLR}_{(\ell-1)}\text{-LR-Product}\left(\widetilde{A}_{ik}, \widetilde{U}_{kj}\right)$
      else
        $\widetilde{A}_{ij} \leftarrow \widetilde{A}_{ij} - \text{BLR}_{(\ell-1)}\text{-BLR}_{(\ell-1)}\text{-Product}\left(\widetilde{A}_{ik}, \widetilde{U}_{kj}\right)$
      end if
    end for
  end for
end for

A.2 Boundedness of the number of non-admissible blocks

In this section, we seek to compute a bound on N_{na}, the maximal number of non-admissible blocks per row and column at any level of the block-hierarchy.

The result is directly derived from Lemma 2 in [4], whose result on BLR matrices we reproduce here.

Lemma 1. *For any BLR matrix, it is possible to build a partition such that the number of non-admissible blocks per row and column is bounded by $N_{na} = O(1)$.*

Proof. See [4], Lemma 2. □

Since this result holds for any BLR matrix, it can be recursively applied to the non-admissible blocks of a ℓ-level matrix, and therefore the result trivially holds at any level of the block hierarchy.

As a consequence, the ℓ-level BLR factorization of a dense matrix clustered into $p = m/b$ blocks requires $O(p^2)$ BLR$_{(\ell-1)}$-LR products and $O(p)$ BLR$_{(\ell-1)}$-BLR$_{(\ell-1)}$ products and triangular solves to be performed.

Now that we have bounded the number of times these kernels are performed, let us bound the cost of performing them once.

A.3 Recursive complexity analysis of the new kernels

Lemma 2. *Let A, B, and U be three ℓ-level BLR matrices of order m and with blocks of rank at most r on any level; in addition, let U be upper triangular. Let also C be a low-rank matrix of order m and rank r. Then, the costs $\mathscr{F}^\ell_{AB}$, $\mathscr{F}^\ell_{AU^{-1}}$, and $\mathscr{F}^\ell_{AC}$ of performing the BLR$_\ell$-BLR$_\ell$ product AB and triangular solve AU^{-1}, and the BLR$_\ell$-LR product AC, respectively, are:*

$$\mathscr{F}^\ell_{AB}(m,r) = \mathscr{F}^\ell_{AU^{-1}}(m,r) = \mathscr{F}^\ell_{ds}(m,r), \tag{51}$$

$$\mathscr{F}^\ell_{AC}(m,r) = O(r) \times \mathscr{S}^\ell_{ds}(m,r), \tag{52}$$

where $\mathscr{F}^\ell_{ds}(m,r)$ and $\mathscr{S}^\ell_{ds}(m,r)$ are given by Theorem 1.

Proof. We proceed by induction. Let us note $b = O(m^x)$ the top level block size of the matrices and $p = m/b = O(m^{1-x})$. Let us also assume that $r = O(m^\alpha)$.

We begin with the initial case $\ell = 1$. The AB product requires $O(p^3)$ LR-LR products of cost $O(br^2)$, $O(p^2)$ LR-FR products of cost $O(b^2r)$, and $O(p)$ FR-FR products of cost $O(b^3)$. Its total cost is therefore $\mathscr{F}^1_{AB} = O(p^3br^2 + p^2b^2r + pb^3) = O(m^{3-2x+2\alpha} + m^{2+\alpha} + m^{1+2x})$, which, for $x = x^* = (1+\alpha)/2$, attains its optimal value $\mathscr{F}^1_{AB}(m,r) = O(m^2r) = \mathscr{F}^1_{ds}(m,r)$. The AU^{-1} triangular solves requires the same computations and therefore has the same asymptotic cost. For the AC product, the fact that $C = \Phi\Psi^T$ is low-rank means that the products can be performed only on Φ, as described in Algorithm 4. Therefore, it only requires $O(p^2)$ LR-LR products and $O(p)$ LR-FR products, which results in a total cost $\mathscr{F}^1_{AC} = O(p^2br^2 + pb^2r) = O(m^{2-x+2\alpha} + m^{1+x+\alpha})$ which, for $x = x^* = (1+\alpha)/2$, attains its optimal value $\mathscr{F}^1_{AC}(m,r) = O(m^{3/2}r^{3/2}) = O(r) \times \mathscr{S}^1_{ds}(m,r)$.

We now assume that the formulas hold for $(\ell-1)$-level BLR matrices and prove them for ℓ-level ones. The AB product requires $O(p^3)$ LR-LR products of cost $O(br^2)$, $O(p^2)$ BLR$_{(\ell-1)}$-LR products of cost $O(r) \times \mathscr{S}^{\ell-1}_{ds}(b,r)$ (by induction), and $O(p)$ BLR$_{(\ell-1)}$-BLR$_{(\ell-1)}$ products of cost $\mathscr{F}^{\ell-1}_{ds}(b,r)$ (also by induction). Using (34) and (35), its total cost is therefore

$$\mathscr{F}^\ell_{AB}(m,x,\alpha) = O(m^{3-2x+2\alpha} + m^{(2\ell+(1-\ell)x+(2\ell-1)\alpha)/\ell} + m^{(\ell+2x+2(\ell-1)\alpha)/\ell}). \tag{53}$$

Taking $x = x^* = (\ell+\alpha)/(\ell+1)$, we obtain $\mathscr{F}^\ell_{AB}(m,r) = \mathscr{F}^\ell_{ds}(m,r)$. The AU^{-1} triangular solve requires the same computations and therefore has the same asymptotic cost. Finally, the AC product requires $O(p^2)$ LR-LR products of cost $O(br^2)$ and $O(p)$ BLR$_{(\ell-1)}$-LR products of cost $O(r) \times \mathscr{S}^{\ell-1}_{ds}(b,r)$, by induction. Therefore, using (34), we obtain the total cost of

$$\mathscr{F}^\ell_{AC}(m,x,\alpha) = O(m^{2-x+2\alpha} + m^{(\ell+x+(2\ell-1)\alpha)/\ell}). \tag{54}$$

Again, for $x = x^* = (\ell+\alpha)/(\ell+1)$, we obtain $\mathscr{F}^\ell_{AC}(m,r) = O(r) \times \mathscr{S}^\ell_{ds}(m,r)$. □

Theorem 2. *The complexity formulas of Theorem 1 established in the weakly-admissible case also hold in the strongly-admissible case.*

Proof. Lemma 1 states that there are only $O(1)$ non-admissible blocks per row and column on any level of the block hierarchy. From this we can already deduce that the storage complexity $\mathscr{S}_{ds}^{\ell}$ is asymptotically the same in the weakly- and strongly-admissible cases. Moreover, using the costs computed in Lemma 2, we can compute the overall cost associated with the new kernels required to be performed in the strongly-admissible case as

$$\mathscr{F}_{\textit{New Kernels}}^{\ell}(b,p,r) = O(p) \times \mathscr{F}_{ds}^{\ell-1}(b,r) + O(p^2 r) \times \mathscr{S}_{ds}^{\ell-1}(b,r) \tag{55}$$

$$= \mathscr{F}_{\textit{Factor}}^{\ell}(b,p,r) + \mathscr{F}_{\textit{Solve}}^{\ell}(b,p,r) \tag{56}$$

which clearly does not asymptotically increase the total, thus proving that $\mathscr{F}_{ds}^{\ell}$ is also unchanged by the strong admissibility condition. □