Recent Advances in Matrix Partitioning for Parallel Computing on Heterogeneous Platforms

Olivier Beaumont, Brett A. Becker, Ashley Deflumere, Lionel Eyraud-Dubois, Thomas Lambert, Alexey Lastovetsky

Recent Advances in Matrix Partitioning for Parallel Computing on Heterogeneous Platforms

Olivier Beaumont, Brett A. Becker, Ashley DeFlumere, Lionel Eyraud-Dubois, Thomas Lambert, and Alexey Lastovetsky

***Abstract*—The problem of partitioning dense matrices into sets of sub-matrices has received increased attention recently and is crucial when considering dense linear algebra and kernels with similar communication patterns on heterogeneous platforms. The problem of load balancing and minimizing communication is traditionally reducible to an optimization problem that involves partitioning a square into rectangles. This problem has been proven to be NP-Complete for an arbitrary number of partitions.**

In this paper, we present recent approaches that relax the restriction that all partitions be rectangles. The first approach uses an original mathematical technique to find the exact optimal partitioning. Due to the complexity of the technique, it has been developed for a small number of partitions only. However, even at a small scale, the optimal partitions found by this approach are often non-rectangular and sometimes non-intuitive.

The second approach is the study of approximate partitioning methods utilizing recursive partitioning algorithms. In particular we use the work on optimal partitioning to improve pre-existing algorithms. In this paper we discuss the different perspectives this approach opens and present two algorithms, SNRPP which is a $\sqrt{\frac{3}{2}}$ approximation, and NRPP which is a $\frac{2}{\sqrt{3}}$ approximation. While sub-optimal, the NRRP approach works for an arbitrary number of partitions. We use the first exact approach to analyse how close to the known optimal solutions the NRRP algorithm is for small numbers of partitions.

I. Introduction

THE problem of partitioning a matrix into a set of sub-matrices has received increased attention in the last few years. This operation is indeed crucial when considering dense linear algebra kernels and other applications with similar communication patterns on heterogeneous platforms. Let us for instance consider dense matrix multiplication based on an algorithm similar to Cannon's, restricted for the sake of simplicity to the multiplication $C = AB$ of two square $n \times n$ matrices A and B. We consider a parallel computation on a shared memory platform, so that the matrices originally reside in the main memory, and the communication costs represent data movement from the main memory to either the caches of the CPUs or the private memory of GPUs. We further assume that the matrices are partitioned into blocks, whose (common) size is chosen so that they are well adapted to all types of resources (typically CPUs and GPUs). In this case, at step k of the algorithm, the outer product of the k-th column of blocks of A and the k-th row of blocks of B is computed. Let us assume that at each step, processor P computes s blocks of matrix C, whose projections along the different axes have respective sizes h and w. Then, the volume of computation for P is proportional to s and the volume of communication required for P to complete its computation is proportional to $h+w$, since it needs to receive the corresponding blocks of both matrices A and B. In order to balance the computing load, each processor should compute a number of blocks of C proportional to its relative speed. In turn, the overall volume of communication is proportional to the sum of the projections of the areas owned by the different processors along the axes. Therefore, in order to minimize the processing time while also minimizing the overall volume of communication, the optimization problem is amenable to the problem of partitioning a square into a set of zones of prescribed area (in order to balance the load) such that the sum of the projections along the two axes is minimized (in order to minimize the communications).

This paper contrasts results of two approaches to this optimization problem when partitioning among small numbers of heterogeneous resources. The first approach is that of Becker and Lastovetsky who proposed that non-rectangular partitionings can be optimal for partitioning between two heterogeneous resources [1], and proven to be optimal among all possible partitionings for two resources (for certain power ratios) by DeFlumere, Lastovetsky and Becker [2]. This approach was extended to three resources [3], and proven (again for certain power ratios) in [4]. The second approach is based on recursive approximation algorithms. Recently, Beaumont, Eyraud-Dubois and Lambert improved the best-known approximation ratio to $\frac{2}{\sqrt{3}} \approx 1.15$ [5] by mixing the work on optimal partitionings for small numbers of partitions and approximation algorithms for rectangle partitioning proposed by Nagamochi et al. [6] and Fügenschuh [7].

II. General Context

A. Related Works

This optimization problem was first introduced by Lastovetsky and Kalinov in [8]. In [9], it was proven that the problem is NP-Complete, and a first approximation algorithm with a bounded ratio of 1.75 was proposed. The assumption of rectangular partitions was relaxed in [1] for partitioning between two resources. This non-rectangular partitioning has perfect load balance and a lower overall volume of communication compared to all rectangular partitionings provided the computational lpower ratio between the two resources

O. Beaumont, L. Eyraud-Dubois, and T. Lambert are with Inria and LaBRI, University of Bordeaux, France

A. Lastovetsky and B.A. Becker are with University College Dublin, Ireland

A. DeFlumere is with Wellesley College, USA

Manuscript received May xx, 2017; revised xx, 2017.

exceeds $3:1$. This was extended to three resources in [3] where a non-rectangular partitioning based on the two partition case was proposed which again has perfect load balancing, and a lower overall volume of communication, again subject to certain power ratio and topology requirements. It is important to note that these partitioning algorithms are not modifications of homogeneous ones; they are designed for heterogeneous scenarios [10]. These algorithms have also been shown to benefit real-world applications [11]. It has also been demonstrated that optimizing the partitioning for non-square matrices is not achievable with straight-forward scaling of the square cases, at least for specific applications such as matrix multiplication [10].

In [2] it was proven that in the case of two resources, the optimal partitioning for any power ratio is of one of two forms, one rectangular and one non-rectangular. A method called the Push technique was first proposed in this work in order to generate a small number of potentially optimal shapes, which can then be analyzed to find the optimal one. The Push technique was extended and used in the study of three resources in [12] in order to identify potential optimal shapes. Six potential shapes were identified in this work. While the optimality of the three resource case has never been mathematically proven, extensive simulations conducted in [13] and [4] failed to find a shape that would outperform these six shapes.

Independently, recursive partitioning algorithms for this optimization problem have been recently proposed. In these, the set of processors is split into two parts at each step, converging on an approximation of the optimal partitioning. Sophisticated proof techniques enabled Nagamochi and Abe [6] to improve the approximation ratio down to 1.25. Recently, Fügenschuh et al. [7] improved this result to 1.15, but under the assumption that if we consider processors in decreasing order of their processing speeds, there is no abrupt change in the performance between any two successive processors. Unfortunately, such an abrupt decrease typically happens when considering more heterogeneous nodes such as those consisting of CPUs and GPUs, therefore restricting Fügenschuh's algorithm to the case of loosely heterogeneous platforms.

Beaumont, Eyraud-Dubois and Lambert [5] sought to keep the best of both worlds by adapting the idea of non-rectangular partitionings proposed by Becker and Lastovetsky and extending it to an arbitrary number of processors by adapting the recursive partitioning algorithm proposed by Nagamochi, which facilitates approximation-ratio proofs. These two ingredients led to an improvement of the approximation ratio down to $\frac{2}{\sqrt{3}} \approx 1.15$. This non-rectangular recursive partitioning algorithm (NRRP), does not require any specific assumption on the relative speed of resources and is therefore applicable to computational power ratios comparable to those of nodes consisting of both regular cores and accelerators.

This partitioning problem can be used as a building block for many dense linear algebra kernels. For instance, it has been extended to LU factorization and other dense linear algebra kernels in [14], [15]. In this case, a block cyclic principle is combined with the initial partitioning in order to obtain 2D cyclic ScaLAPACK solutions [16], where the load is balanced throughout the whole computation. These partitionings have also been adapted to distributed hierarchical and highly heterogeneous platforms in [17], where the partitioning is applied at two levels (intra-node and inter-node), based on sophisticated performance models. The same partitioning has also been extended to finite-difference time-domain (FDTD) methods to obtain numerical solutions of Maxwell's equations in [18]. The extension to more dynamic settings has also been considered in [19]. In this case, the partitioning problem can be used in order to provide an initial static partitioning algorithm that can be modified in order to dynamically maintain load balancing. Recently, in order to cope with resource heterogeneity and the difficulty in building optimal schedules, the use of dynamic runtime schedulers has been proposed, such as StarPU [20], StarSs [21], QUARK [22] or PaRSEC [23]. At runtime, the scheduler takes the scheduling and allocation decisions based on the set of ready tasks (tasks for which data and control dependences have been solved), on the availability of the resources (estimated using expecting processing and communication times), and on the actual location of input data. The comparison between static scheduling strategies (such as the one proposed in this paper) and runtime scheduling strategies has been recently considered in [24], where the analysis of the behavior of static, dynamic, and hybrid strategies highlights the benefits of introducing more static knowledge and allocation decisions in runtime libraries.

The results of all of these papers are based on the partitioning problem considered in this paper and can therefore directly benefit from an improvement in the performance and approximation ratio.

B. Notations and Problem Statement

In this section, we define the notations that will be used in the rest of this paper and we present the formal version of the optimization problem that corresponds to enforcing a perfect load balancing while minimizing the amount of communications.

The problem comes in several flavors depending on the constraints imposed to the partition. In this paper, in order to have a solution that is independent of the matrix size (but only dependent on the relative speeds of the resources), we will consider the continuous version of the problem, where the goal is to split the unit square $[0,1] \times [0,1]$. In general, such a solution has to be rounded for a given matrix size. How to perform these roundings optimally has been considered in [24] and we rely on these results. Another possible restriction is related to the shape of the area Z_l allocated to processor P_l. In many of the above mentioned works, it has been assumed that Z_l had to be shaped as a rectangle [9], [6], [7] because this was considered as a requirement for a simple and efficient implementation. Recent task-based implementations allow this constraint to be relaxed, and it is thus possible to search for general (non-rectangular) partitionings. Such partitionings have been proposed in [13], [3], and it has been proven that it is possible to derive significantly better partitionings when removing the rectangular constraint. Therefore, in this paper,

we do not impose that the areas allocated to the processors be shaped as rectangles.

We thus consider the unit square $S = [0,1] \times [0,1]$. Let Z denote a zone (a subset of S) included in the unit square. We denote by $s(Z)$ its area (formally $\iint_Z dxdy$) and by $R(Z)$ its covering rectangle, *i.e.* the Cartesian product of the projections of Z along both dimensions. If $R(Z) = [x_1, x_2] \times [y_1, y_2]$, then we define the height of Z by $h(Z) = x_2 - x_1$ and the width of Z by $w(Z) = y_2 - y_1$. Finally, let us define $p(Z) = h(Z) + w(Z)$, the half-perimeter of $R(Z)$ and $\rho(Z) = \frac{\max(h(Z), w(Z))}{\min(h(Z), w(Z))}$ its aspect ratio.

We consider the following problem :

Problem 1 (PERI-SUM)**.** *Given a set of n positive rational numbers $\{s_1, \ldots, s_n\}$ such that $\sum s_k = 1$, and the square $S = [0,1] \times [0,1]$, find for each s_k a zone $Z_k \subseteq S$ such that the area of Z_k is s_k, $\bigcup Z_k = S$, and such that $\sum p(Z_k)$ is minimized.*

In the following, we denote $\sum p(Z_k)$ as $c(Z_1, \ldots, Z_n)$ and its optimal value as c_{opt}. A lower bound has been proposed by Ballard et al. in [25], that comes from an application of the Loomis-Whitney inequality. This lower bound simply states that the perimeter of a zone Z_k of given area $s(Z_k)$ is minimal when the zone is shaped as a square.

$$c(Z_k) \geq 2\sqrt{s(Z_k)} \tag{1}$$

This provides a lower bound on the total cost for any instance, $c_{opt} \geq 2\sum_k \sqrt{s_k}$. This bound is reached only when all zones are squares, which may not be feasible depending on the instance (consider for instance the case of two identical zones of area $1/2$), and hence this lower bound is in general optimistic.

C. Complexity Results

A variant of the decision problem associated to this optimization problem has been proved to be NP-Complete. More specifically, it has been proven in [9] that deciding whether $S = [0,1] \times [0,1]$ can be partitioned into squares of respective areas $\{s_1, \ldots, s_p\}$ is NP-Complete.

Since squares are the best possible shapes for the zones irrespective to any additional constraint, this complexity result automatically translates to PERI-SUM, which is thus an NP-Complete problem. For this reason, the rest of the paper is devoted to finding algorithms in two directions: either exact algorithms for a small number of processors, or approximation algorithms for the general case.

D. Theoretical comparison with block-cyclic approaches

Distributed matrix multiplication implementations often use classic partitioning schemes, such as 2D cyclic or block-cyclic decompositions. In such decompositions, processors are arranged into a $p \times q$ grid: all processors on the same row receive the same rows of matrix A, and all processors on the same column receive the same columns of matrix B. These decompositions can only be used in the (very) special case of homogeneous processors, since they are inherently designed to provide each processor with the same amount of computation. Designing block-cyclic decompositions that would best adapt to heterogeneous computing speeds is itself a difficult problem. Since we focus on heterogeneous settings, we can not compare our results directly to block-cyclic decompositions.

For homogeneous processors, since each element of matrix A is sent to p processors, and each element of matrix B is sent to q processors, the total communication cost of a block-cyclic decomposition is $p+q$. When $p = q$ $(= \sqrt{n})$, the cost is $2\sqrt{n}$, which is exactly equal to the lower bound $2\sum_{i=1}^{n} \frac{1}{\sqrt{n}}$. However, if p and q are very different, the ratio to the lower bound can be very large. The worst case is reached when one of p or q is equal to 1, where the communication cost is $n + 1$, which yields an unbounded ratio with respect to the lower bound.

III. Optimal Partitionings

Traditionally many applications such as matrix multiplication (which is generalizable to many other applications of interest) are performed in parallel by dividing matrices into rectangular partitionings. We challenge this approach. In this section we study the optimality of partition shapes for a small number of partitions. Traditionally the problem of finding the optimal shape is reduced to PERI-SUM, the problem of finding the optimal partitioning that minimizes the sum of half perimeters (SHP), which, as discussed in Section II, is equivalent to the projections of the areas owned by the different processors along the axes. This is the simplest mathematical formulation of this problem. Our motivation is that when we balance the load (which is normally trivial) and minimize the SHP, we minimize the amount of data moved between partitions, and thus the required communication for a given application.

For small numbers of heterogeneous processors, potentially optimal partition shapes can be determined using the Push Technique [2], [12].

In the following, we denote by P the fastest processor, by Q the second fastest, and when applicable, by R the slowest one ($s_P \geq s_Q \geq s_R$).

A. Two Heterogeneous Processors

Two partition shapes shown in Figure 1 are identified as the optimal two processor partitioning in [2].

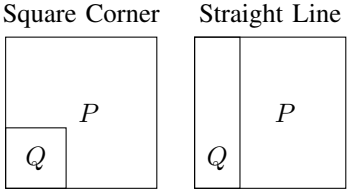

Figure 1: The optimal partitions with two processors.

Theorem 1. *For two heterogeneous processors, the data partitioning with the optimal total volume of communication*

will be the Square Corner when the ratio of computational power between the two processors is greater than three.

Proof. The fact that the two partitions of Figure 1 are dominant is a corollary of Theorem 2, proven below. We can compute the cost of each partition:

- *Square Corner*. The cost for P is 2, and the cost for Q is $2\sqrt{s_Q}$, for a total cost of $c_{SC} = 2 + 2\sqrt{s_Q}$.
- *Straight Line*. The cost for P is $1 + s_P$, and the cost for Q is $1 + s_Q$, with $s_P + s_Q = 1$, for a total cost of $c_{SL} = 3$. Alternatively, we can see that one dimension is sent to both processors P and Q, while the other is shared among them, hence the cost of 3.

It is clear that $c_{SC} < c_{SL}$ if and only if $\sqrt{s_Q} < \frac{1}{2}$, which is equivalent to $s_Q < \frac{1}{4}$. □

B. Three Heterogeneous Processors

In [12], the Push technique was used to propose six candidate partition shapes for consideration to be the optimal three processor partitioning of a square shown in Figure 2. The optimality of these shapes has never been proven however. What was proven is that the Rectangle Corner and the Straight Line both have a higher communication cost than the Block Rectangle [13]. It has also become evident that the L Rectangle and the Block Rectangle are topologically equivalent.

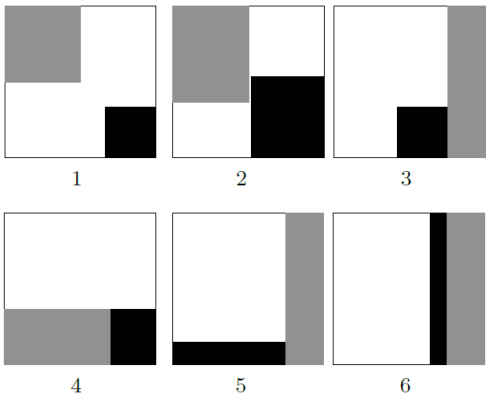

Figure 2: The candidate partition shapes identified as potentially optimal three-processor shapes in [12]. (1) Square Corner (2) Rectangle Corner (3) Square Rectangle (4) Block Rectangle (5) L Rectangle (6) Straight Line

In this paper, we prove that the four shapes shown in Figure 3 are sufficient to optimally partition an arbitrary rectangle. We also prove that three of these shapes - the Square Corner, the Square Rectangle, and the Block Rectangle, are sufficient to optimally partition a square.

Theorem 2. *The optimal data partition shape to split a rectangle between three processors is always one of the four shapes of Figure 3: the Square Corner, the Square Rectangle, the Block Rectangle, or the Straight Line.*

Proof. Without loss of generality, we assume that the rectangle to partition has dimension $a \times b$, with $a \leq b$, and a is the size of the vertical axis. We first compute the cost of each solution (note that the first two might not be feasible):

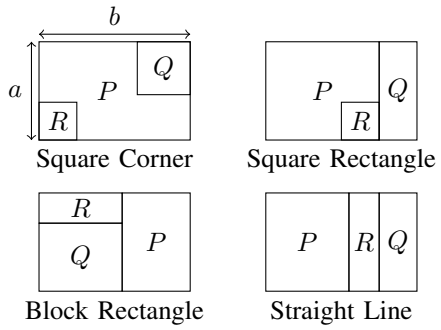

Figure 3: The optimal data partitions for a rectangle with three processors.

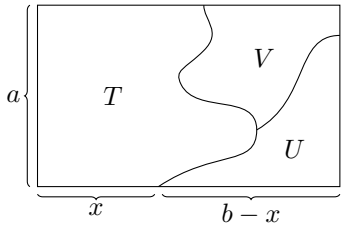

Figure 4: Notation for the case where one processor holds a complete side

- *Square Corner* $c_{SC} = a + b + 2(\sqrt{s_Q} + \sqrt{s_R})$
- *Square Rectangle* $c_{SR} = 2a + b + 2\sqrt{s_R}$
- *Block Rectangle* $c_{BR} = 2a + 2b - \frac{s_P}{a}$
- *Straight Line* $c_{SL} = 3a + b$

To prove Theorem 2, we consider any valid partition $\mathcal{S}$, and we prove that the cost C of $\mathcal{S}$ is at least as much as the cost of one of the proposed shapes. The proof is split into 5 different cases, depending on the number of resources that hold a complete row or column in $\mathcal{S}$. In the following, we say that a processor holds a complete row (resp. column) if the projection of its allocated area on the horizontal (resp. vertical) axis is equal to the complete side of the rectangle.

(i) No processor holds a complete side in $\mathcal{S}$.
In this case, each data is sent to at least 2 resources, so that the overall size of the projections is at least $2a + 2b$. Since the Straight Line position in Figure 3 is always feasible and has cost $3a + b \leq 2a + 2b$, $\mathcal{S}$ is dominated.

(ii) Exactly one processor T holds a complete side in $\mathcal{S}$, but not both sides.
Let us first assume that processor T holds a complete column, and let us denote by x the length of the part of the horizontal axis on which only the area allocated to T is projected (*i.e.*, the amount of horizontal data that is sent only to T, see Figure 4). Then, the other $b - x$ columns are sent to at least 2 resources (otherwise another resource would hold a complete side), and each row is sent to at least 2 resources as well (T and at least another one) so that the cost of $\mathcal{S}$ is at least $2a + x + 2 * (b - x) = 2b + 2a - x$. By construction $x \leq \frac{s_T}{a}$, and the cost of the

Block Rectangle solution is $2b+2a-\frac{s_P}{a}$, with $s_P \geq s_T$. Hence $\mathcal{S}$ is dominated. Similarly, if T holds a complete row, the cost of $\mathcal{S}$ is at least $2a+2b-x$, with $x \leq \frac{s_T}{b}$, and is thus also dominated by Block Rectangle since $a \leq b$.

(iii) Exactly one processor T holds a complete row and a complete column in $\mathcal{S}$.
In this case, T contributes for at least $a+b$ to the cost of $\mathcal{S}$, and therefore $C \geq a+b+2\sqrt{s_U}+2\sqrt{s_V}$, where U and V are the two other processors.
- If $\sqrt{s_U}+\sqrt{s_V} > a$, then $C > 3a+b$ and $\mathcal{S}$ is again dominated.
- If $\sqrt{s_U}+\sqrt{s_V} < a$, then the Square Corner solution of Figure 3 is feasible, and has cost exactly $a+b+2\sqrt{s_Q}+2\sqrt{s_R}$. Since Q and R are the two smallest size values, this solution dominates $\mathcal{S}$.

(iv) Exactly 2 resources T and U hold a complete side in $\mathcal{S}$. For example, let us assume that they both hold a complete column. Since the remaining resource V does not hold a complete column, each data on the horizontal axis is sent at least to either T or U. Then, the contribution of T and U to C is at least $2a+b$ and therefore $C \geq 2a+b+2\sqrt{s_V}$. If both resources hold a complete row instead, we obtain similarly that $C \geq 2b+a+2\sqrt{s_V} \geq 2a+b+2\sqrt{s_V}$.
- If $\sqrt{s_V} > a$, then $C > 4a+b$. Since the Straight Line solution has cost $3a+b$, this implies that $\mathcal{S}$ is dominated (by Straight Line solution).
- If $\sqrt{s_V} < a$, then solution Square Rectangle from Figure 3 is feasible, and has cost $2a+b+2\sqrt{s_R}$. Since $s_V \geq s_S$, $\mathcal{S}$ is dominated.

(v) All 3 resources hold a complete side in $\mathcal{S}$. If they hold a complete column, then $C \geq 3a+b$, otherwise $C \geq 3b+a \geq 3a+b$. Then, $\mathcal{S}$ is dominated by the Straight Line solution.

□

It is interesting to note that the Straight Line solution is never optimal when partitioning a square (*i.e.*, when $a = b$), since it is always dominated by Block Rectangle. Of the remaining three, the first two are non-rectangular, but non-rectangular in different ways which make each suited to a different type of heterogeneous distribution of computational power. In general, these results show that, for a square partitioning (see Figure 11 in Section V):

- Square Corner is the optimal shape for heterogeneous systems with a single fast processor, and two relatively slow processors.
- Square Rectangle is the optimal shape for heterogeneous systems with two fast processors, and a single relatively slow processor.
- Block Rectangle is the optimal shape for heterogeneous systems with a fast, medium, and slow processor, as well as relatively homogeneous systems.

IV. Non-Rectangular Recursive Partitioning (NRRP)

Extending the Push Technique to an arbitrary number of partitions is not trivial, and the challenge increases significantly for each additional partition (the three partition case was much more challenging than the two partition case). This creates a demand for approximation algorithms for arbitrary numbers of partitions. We are inspired by the recursive results of Nagamochi and Abe [6] whose recursive algorithm reduced the approximation ratio to 1.25 without significantly restricting the problem. We develop a recursive algorithm that guarantees not to be more than $\frac{2}{\sqrt{3}}$ from optimal. Our motivation is to be able to compute fast a partitioning that is known to be within a small factor from the optimal for any arbitrary number of partitions.

This section is devoted to approximation algorithms for PERI-SUM. We present several increasingly complex algorithms, to finish with NRRP (Non-Rectangular Recursive Partitioning), whose approximation ratio is below $\frac{2}{\sqrt{3}}$.

A. Rectangular Recursive Partitioning

Algorithm 1: RRP$(R, \{s_1, \ldots, s_n\})$

Input: A rectangle R, a set of positive values $\{s_1, \ldots, s_n\}$ such that $\sum s_i = s(R)$ and $s_1 \leq s_2 \leq \ldots \leq s_n$

Output: For each $1 \leq i \leq n$, a rectangle R_i such that $s(R_i) = s_i$ and $\bigcup R_i = R$

```
if n = 1 then
    return R
else
    ρ = ρ(R) ;
    k = the smallest k such that Σ_{i=1}^{k} s_i ≥ s/3 ;
    s' = Σ_{i=1}^{k} s_i ;
    Cut R in R_1, R_2 according to its greatest dimension
      and with s(R_1) = s' ;
    return
      RRP(R_1, {s_1, ..., s_k}) + RRP(R_2, {s_{k+1}, ..., s_n})
```

The first algorithm is called RRP (Rectangular Recursive Partitioning, see Algorithm 1) and is inspired by the seminal work of Nagamochi et al. [6]. The basic idea is to use the divide and conquer paradigm by splitting the current rectangle in two sub-rectangles at each step of the algorithm, and assign to each part a sublist of the list of processors. The cut is made so as to have, if possible, at least a third of the total area on each part. For this purpose, the areas are sorted in increasing order and aggregated one by one until one third of the total is reached. Lemma 3 shows that if at least one item remains, then this method ensures that both parts contain at least one third of the total area.

Lemma 3. *Let $\{s_1, \ldots, s_n\}$ be a set of positive values such that $s_1 \leq \ldots \leq s_n$ and k be the smallest k such that $\sum_{i=1}^{k} s_i \geq \frac{s}{3}$. Then, if $k < n$, $\sum_{i=k+1}^{n} s_i \geq \frac{s}{3}$.*

Proof. By definition of k, $\sum_{i=1}^{k-1} s_i < \frac{s}{3}$. Let us assume that $\sum_{k+1}^{n} s_i < \frac{s}{3}$ for the search of a contradiction. In this case we obtain that $s_k \geq \frac{s}{3}$. As $s_{k+1} \leq \sum_{i=k+1}^{n} s_i < \frac{s}{3}$, we have $s_k > s_{k+1}$ which is a contradiction with the hypothesis $s_1 \leq \ldots \leq s_n$. □

Otherwise, if $k = n$, this means that $\sum_{i=1}^{n-1} s_i < \frac{s}{3}$, and hence s_n is significantly greater than the other values. This is actually the pathological case on which the following algorithms improve. If we disregard this case for a moment, then the algorithm is quite effective and simply using Lemma 4 leads directly to an approximation ratio of $\frac{2}{\sqrt{3}}$, as Fügenschuh et al point out [7]. Note that the proof of Lemma 4 relies on the lower bound from (1) and does not compare directly to the true optimal value.

Lemma 4. *Let R be a rectangle. Then:*

$$\frac{p(R)}{2\sqrt{s(R)}} = \frac{1+\rho(R)}{2\sqrt{\rho(R)}}.$$

In particular, if $\rho(R) \leq 3$ then:

$$\frac{p(R)}{2\sqrt{s(R)}} \leq \frac{2}{\sqrt{3}}.$$

Proof. Let us assume without loss of generality that $h = h(R) \leq w = w(R)$ and denote $\rho = \rho(R) = \frac{w}{h}$. Then $p(R) = h+w = h(1+\rho)$. In addition $s = s(R) = hw = \rho h^2$. Therefore

$$\frac{w+h}{2\sqrt{s}} = \frac{h(1+\rho)}{2\sqrt{\rho h^2}} = \frac{1+\rho}{2\sqrt{\rho}}.$$

Furthermore, one can prove that $x \mapsto \frac{1+x}{2\sqrt{x}}$ is an increasing function on $[1,3]$ and then, for $\rho \leq 3$:

$$\frac{w+h}{2\sqrt{s}} \leq \frac{1+3}{2\sqrt{3}} = \frac{2}{\sqrt{3}}.$$

□

However, the pathological cases imply that the actual approximation ratio of RRP is at least $\frac{3}{2}$. Indeed, let us consider an instance with two values ϵ and $1-\epsilon$. The optimal partitioning is depicted on the left of Figure 5, the communication cost is $2\sqrt{\epsilon}$ for the zone of area ϵ and 2 for the other zone, which yields a total of $2(1+\sqrt{\epsilon})$. The partitioning returned by RRP is shown on the right of Figure 5. In this case, the communication cost is $1+\epsilon$ for the smaller zone and $2-\epsilon$ for the larger one, which gives a total communication cost of 3. It results that $\frac{Cost(RRP)}{C_{opt}} = \frac{3}{2(1+\epsilon)} \underset{\epsilon \to 0}{\longrightarrow} \frac{3}{2}$. In their work, Nagamochi et al. prove that RRP is actually a $\frac{5}{4}$-approximation algorithm for the variant of the problem where all the zones have to be rectangles (and indeed in this case the partitioning on the right of Figure 5 is the only possible partitioning and therefore also the optimal one).

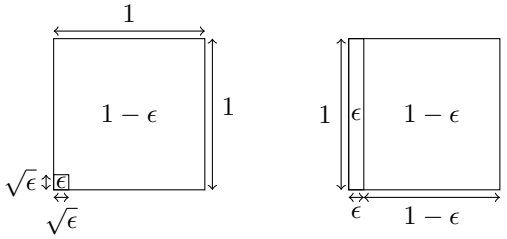

Figure 5: Two partitions of a square with the instance $\{\epsilon, 1-\epsilon\}$.

B. Simple Non-Rectangular Recursive Partitioning

As a first step to improve RRP, we use the insight from Section III: whenever a value is significantly bigger than the others, it is best to avoid splitting into two rectangles. We thus slightly adapt the algorithm and obtain two subroutines.

Guillotine: The first one is the Guillotine routine, depicted in Figure 6 (left). It is the main ingredient of RRP. Given a composed zone R and a rational number $\alpha \in [0,1]$, $Guillotine(R,\alpha)$ splits R along the largest dimension into two rectangles of respective areas $\alpha s(R)$ and $(1-\alpha)s(R)$.

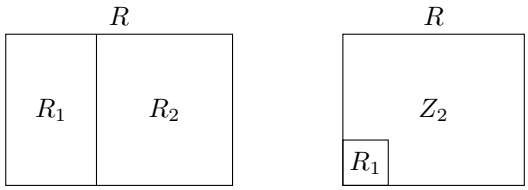

Figure 6: An illustration of the Guillotine (left) and Square (right) routines.

Square: The second routine is the Square routine, depicted in Figure 6 (right). Given a rectangle R and a rational number $\alpha \in [0,1]$, $Square(R,\alpha)$ returns a square R_1 of area $\alpha s(R)$ and a zone Z_2 which corresponds to the initial rectangle R punched by square R_1. The covering rectangle of Z_2 is R and Z_2 will always be used to host a simple zone.

With these two subroutines we propose an improved algorithm: SNRRP (Simple Non-Rectangular Recursive Partitioning), given in Algorithm 2. The basic idea is the following: according to an appropriate condition, the current rectangle is either split into two well-shaped rectangles (lines 7-9), or into a square and its complement, the latter being assigned to a single processor (lines 10-12).

Algorithm 2: SNRRP$(R, \{s_1, \ldots, s_n\})$

Input: A rectangle R, a set of values $\{s_1, \ldots, s_n\}$ such that $\sum s_i = s(R)$ and $s_1 \leq s_2 \leq \ldots \leq s_n$
Output: For each $1 \leq i \leq n$, a zone Z_i such that $s(Z_i) = s_i$ and $\bigcup Z_i = R$

1 **if** $n = 1$ **then**
2 **return** R
3 **else**
4 $\rho = \rho(R)$;
5 $k =$ the smallest k such that $\sum_{i=1}^{k} s_i \geq \frac{s}{3\rho}$;
6 $s' = \sum_{i=1}^{k} s_i$;
7 **if** $k < n$ **then**
8 $R_1, R_2 =$ `Guillotine`$(R, s'/s)$;
9 **return** `SNRRP`$(R_1, \{s_1, \ldots, s_k\}) +$ `SNRRP`$(R_2, \{s_{k+1}, \ldots, s_n\})$
10 **else**
11 $R_1, Z_2 =$ `Square`$(R, (s-s_n)/s)$;
12 `SNRRP`$(R_1, \{s_1, \ldots, s_{n-1}\}) + Z_2$

This algorithm can be proven to be a $\sqrt{\frac{3}{2}}$-approximation ($\sqrt{\frac{3}{2}} \simeq 1.22$) and thus be seen as a significant improvement

of RRP. The proof relies on one major invariant (that also ensures the correctness of the algorithm): each rectangle on which the function is called has an aspect ratio below 3, as is steted in Theorem 5.

Theorem 5 (Correctness)**.** *When executing SNRRP$(R, \{s_1, \ldots, s_n\})$ with $\rho(R) \leq 3$, all the recursive calls to SNRRP$(R', \{s'_1, \ldots, s'_k\})$ are performed on a rectangle area R' such that $\rho(R') \leq 3$.*

Proof. Recursive calls to SNRRP are made on two kinds of rectangles, those produced by Guillotine (line 8) and the square produced by Square (line 11). In the second case the aspect ratio is trivially below 3. In the first case, let s and s' be as described in Algorithm 2, $\alpha = s'/s$ and let $R_1, R_2 = Guillotine(R, \alpha)$. We have to prove that $\rho(R_1) \leq 3$ and $\rho(R_2) \leq 3$. By hypothesis we know that $\alpha \geq \frac{1}{3\rho(R)}$.

Let us assume without loss of generality that $h = h(R) \leq w = w(R)$ and denote $\rho = \rho(R) = \frac{w}{h}$. By definition $h(R_1) = h(R)$ and $w(R_1) = \alpha w$. Therefore $\rho(R_1) = \max(\frac{\alpha w}{h}, \frac{h}{\alpha w})$. We have $\frac{\alpha w}{h} \leq \frac{w}{h} = \rho \leq 3$. In addition, since $\alpha \geq \frac{1}{3\rho}$, $\frac{h}{\alpha w} \leq \frac{3\rho h}{w} = 3$. Hence, if $\alpha \geq \frac{1}{3\rho(R)}$ then $\rho(R_1) \leq 3$.

To prove that $\rho(R_2) \leq 3$ we can slightly adapt Lemma 3 and show that $1 - \alpha \geq \frac{1}{3\rho(R)}$. Then, similar arguments prove that $\rho(R_2) \leq 3$ and yield the proof of the Theorem. □

To achieve the approximation proof, let us first note that the zones returned by the algorithm are produced at line 2 and 12. In the first case, the zone is a rectangle with an aspect ratio below 3 (Theorem 5) and Lemma 4 proves that its half-perimeter is below $\frac{2}{\sqrt{3}}$ times the lower-bound (and $\frac{2}{\sqrt{3}} \leq \sqrt{\frac{3}{2}}$). In the case of line 12, the resulting zone is no longer rectangular and an additional lemma is required.

Lemma 6. *Let R be a rectangle such that $\rho(R) \leq 3$, $\alpha \in [0, 1]$ and $R', Z = Square(R, \alpha)$. If $\alpha \leq \frac{1}{3\rho(R)}$ then $\frac{p(Z)}{2\sqrt{s(Z)}} \leq \sqrt{\frac{3}{2}}$.*

Proof. Let us denote $\rho(Z) = \rho$ and $s(Z) = s$ and let us suppose that $h = h(R) \leq w = w(R)$ without loss of generality. Note that the covering rectangle of Z is R and therefore $w = w(Z)$ and $h = h(Z)$. In this case $w = \rho h$, which implies $p(Z) = (1 + \rho)h$. By definition of Square, $s = (1 - \alpha)s(R)$ and $s(R) = hw = \rho h^2$. Therefore $s = (1 - \alpha)\rho h^2 \geq (1 - \frac{1}{3\rho(Z)})\rho h^2$ and hence

$$\frac{p(Z)}{2\sqrt{S(Z)}} \leq \frac{(1+\rho)h}{2\sqrt{(1-1/3\rho)\rho h^2}} \leq \frac{\sqrt{3}(1+\rho)}{2\sqrt{(3\rho-1)}}.$$

One can prove that the function $x \longmapsto \frac{\sqrt{3}(1+x)}{2\sqrt{3x-1}}$ reaches its maximum on $[1, 3]$ for $x = 1$ and $x = 3$ and this maximum is $\sqrt{\frac{3}{2}}$. Thus, $\frac{p(Z)}{2\sqrt{S(Z)}} \leq \sqrt{\frac{3}{2}}$. □

Lemma 6 covers the second case of zone creation. Indeed, the hypothesis of the lemma on the aspect ratio of R is ensured by Theorem 5 and the hypothesis on α comes from the fact that, in line 11 of Algorithm 2, $s - s_n = \sum_{i=1}^{n-1} s_i = \sum_{i=1}^{k-1} s_i < \frac{s}{3\rho(R)}$ by definition of k in line 5.

All of the above proves that if $\{Z_1, \ldots, Z_n\} = SNRRP(R, \{s_1, \ldots, s_n\})$, then for all $i \in [1, n]$, $\frac{p(Z_i)}{2\sqrt{s_i}} \leq \sqrt{\frac{3}{2}}$. Therefore, $\frac{\sum_{i=1}^n p(Z_i)}{\sum_{i=1}^n 2\sqrt{s_i}} \leq \sqrt{\frac{3}{2}}$. Since $\sum_{i=1}^n 2\sqrt{s_i}$ is the lower bound from (1), this achieves the proof Theorem 7.

Theorem 7. *SNRRP is a $\sqrt{\frac{3}{2}}$-approximation for PERI-SUM.*

Note that there are cases where Algorithm 2 returns a partition such that the ratio between the sum of perimeters and the lower bound based on Equation 1 is indeed $\sqrt{\frac{3}{2}}$. Let us define for $1 < k \leq n$, $s_k = 2/3^{n-k+1}$ and $s_1 = 1/3^{n-1}$. One can notice that $s_k = \frac{2}{3}\sum_{i=1}^{k} s_i$. Hence each step of $SNRRP([0, 1]^2, \{s_1, \ldots, s_n\})$ corresponds to the case of line 11-12 and therefore the Square routine is called. In addition, this corresponds the extremal position of the case of line 11-12 and we can notice in the proof of Lemma 6 that in this case the bound is tight. Hence, if $Z_1, \ldots, Z_n = NRRP([0, 1]^2, \{s_1, \ldots, s_n\})$, $p(Z_1) = 2\sqrt{s(Z_1)}$ and for $k > 1$, $p(Z_k) = 2\sqrt{\frac{3}{2}}\sqrt{s(Z_k)}$. Thus,

$$\lim_{n\to\infty} \frac{\sum_{i=1}^n p(Z_i)}{2\sum_{i=1}^n \sqrt{s(Z_i)}} = \sqrt{\frac{3}{2}}$$

An illustration of this case is depicted in Figure 7.

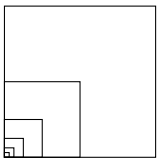

Figure 7: Worst case scenario for SNRRP.

C. Non-Rectangular Recursive Partitioning

Further improvements over SNRRP and its approximation ratio can be obtained by changing the invariant. In both RRP and SNRRP, the aim is to obtain a rectangle with an aspect ratio below 3. This can be generalized by aiming for a rectangle with an aspect ratio below μ where μ is a parameter whose value will be determined later. The objective is to use values of $\mu < 3$, so that the zones produced are as close to squares as possible; the algorithm NRRP that we propose uses $\mu = \frac{5}{2}$. With values of μ different from 3, the proofs change at two places.

1) Aspect ratio guarantees: The first aspect of the proof of SNRRPis Lemma 3 which ensures the invariant that all recursive calls are made on rectangles with aspect ratio below 3. This lemma can be generalized for $\mu \geq 3$ and can be rewritten as Lemma 8.

Lemma 8. *For $\mu \geq 3$, let $\{s_1, \ldots, s_n\}$ be a set of positive values which sum to s such that $s_1 \leq \ldots \leq s_n$, and let k be the smallest k such that $\sum_{i=1}^k s_i \geq \frac{s}{\mu}$. Then, if $k < n$, $\sum_{i=k+1}^n s_i \geq \frac{s}{\mu}$.*

However, in the case $\mu < 3$, the lemma has to be significantly weakened, as expressed in Lemma 9.

Lemma 9. *Let $p > 1$ be an integer and let μ be in $[\frac{2p+1}{p}, \frac{2p-1}{p-1}[$. Let $\{s_1, \ldots, s_n\}$ be a set of positive values such that $s_1 \leq \ldots \leq s_n$ and k be the smallest k such that $\sum_{i=1}^{k} s_i \geq \frac{s}{\mu}$. Then, if $k \leq n-p$, $\sum_{i=k+1}^{n} s_i \geq \frac{s}{\mu}$.*

Proof. By definition of k, $\sum_{i=1}^{k-1} s_i < \frac{s}{\mu}$. Let us assume $\sum_{i=k+1}^{n} s_i < \frac{s}{\mu}$ for the search of a contradiction. In this case we obtain that $s_k \geq \frac{\mu-2}{\mu} s$. In addition $\sum_{i=k+1}^{n} s_i \geq \sum_{i=k+1}^{n} s_{k+1} = (n-k)s_{k+1}$. Therefore, $s_{k+1} < \frac{s}{(n-k)\mu}$. From the assumption $k \leq n-p$ we get that $s_{k+1} < \frac{s}{p\mu}$. Furthermore, $\mu - 2 \geq \frac{2p+1}{p} - 2 \geq \frac{1}{p}$. Hence $s_k \geq \frac{s}{p\mu} > s_{k+1}$, what is a contradiction with $s_1 \leq \ldots \leq s_n$. □

Both lemmas provide a sufficient condition for the possibility to split the list of values such that each part sums to a fraction of at least $\frac{1}{\mu}$ of the total, allowing to satisfy the invariant when performing a recursive call on each sublist. When the condition is not met, the implication in the case $\mu \geq 3$ is that there exists a single large value $s_n \geq s(1 - \frac{1}{\mu})$, for which a large zone can be accommodated with low communication cost by the Square routine. When $\mu < 3$ however, this guarantee is weakened to the existence of at most $p = \left\lfloor \frac{1}{\mu-2} \right\rfloor$ values whose sum is at least $s(1 - \frac{1}{\mu})$. It is thus necessary to design routines that can accommodate all p zones with low communication cost.

For NRRP, where we choose $\mu = \frac{5}{2}$ (and thus $p = 2$), we introduce the routine Tripartition, shown on Figure 8. This routine is used when (with the notations of Lemma 9) $k = n-1$, and divides the current rectangle R into two terminal zones Z_1 and Z_3 which are assigned to the two large remaining values, and one rectangle R_1 on which the recursive call is made.

Tripartition : The routine $Tripartition(R, \alpha, \beta)$ returns three rectangles R_1, Z_2 and Z_3 of respective areas $\alpha s(R)$, $\beta s(R)$ and $(1-\alpha-\beta)s(R)$ as depicted on Figure 8. In NRRP, the recursive call is thus only made on R_1 and we only need to prove $\rho(R_1) \leq \frac{5}{2}$ as an invariant on the aspect ratios of rectangles.

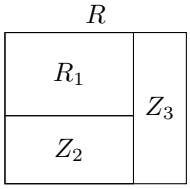

Figure 8: An illustration of the Tripartition routine.

2) Approximation ratios: Changing the value of μ has also an effect on the approximation ratios for the individual zones returned by the algorithm. For the case where the algorithm returns a rectangle, the generalized version of Lemma 4 is stated in Lemma 10, with an identical proof.

Lemma 10. *Let R be a rectangle. If $\rho(R) \leq \mu$ then:*

$$\frac{p(R)}{2\sqrt{s(R)}} \leq \frac{\mu+1}{2\sqrt{\mu}}.$$

This function is increasing for the interesting values of μ, and thus decreasing μ improves the approximation ratio in the case where all the produced zones are shaped as rectangles. However, as shown above, this is not the limiting worst-case; indeed, $\frac{2}{\sqrt{3}}$ ($\frac{\mu+1}{2\sqrt{\mu}}$ for $\mu = 3$) is smaller than $\sqrt{\frac{3}{2}}$. Hence one could expect that slightly increasing this part of the bound with a larger value of μ would allow to lower the approximation ratios obtained for zones returned as complements of a square, given in Lemma 6. However the generalized version, Lemma 11, does not allow this.

Lemma 11. *Let R be a rectangle such that $\rho(R) \leq \mu$, $\alpha \in [0,1]$ and $R', Z = Square(R, \alpha)$. If $\alpha \leq \frac{1}{\mu\rho(R)}$ then $\frac{p(Z)}{2\sqrt{s(Z)}} \leq \max(\sqrt{\frac{\mu}{\mu-1}}, \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}})$.*

Proof. Let us denote $\rho(Z) = \rho$ and $s(Z) = s$ and let us suppose that $h = h(R) \leq w = w(R)$ without loss of generality. Note that the covering rectangle of Z is R and therefore $w = w(Z)$ and $h = h(Z)$. In this case $w = \rho h$, what implies $p(Z) = (1+\rho)h$. By definition of Square, $s = (1-\alpha)s(R)$ and $s(R) = hw = \rho h^2$. Therefore $s = (1-\alpha)\rho h^2 \geq (1 - \frac{1}{\mu\rho(Z)})\rho h^2$ and hence

$$\begin{aligned}\frac{p(Z)}{2\sqrt{S(Z)}} &\leq \frac{(1+\rho)h}{2\sqrt{(1-1/\mu\rho)\rho h^2}} \\ &\leq \frac{\sqrt{\mu}(1+\rho)}{2\sqrt{(\mu\rho-1)}}.\end{aligned}$$

One can prove that the function $x \longmapsto \frac{\sqrt{\mu}(1+x)}{2\sqrt{\mu x-1}}$ is decreasing on $[1, 1+\frac{2}{\mu}]$ and increasing on $[1+\frac{2}{\mu}, \mu]$ and then $\frac{\sqrt{\mu}(1+x)}{2\sqrt{\mu x-1}} \leq \max(\sqrt{\frac{\mu}{\mu-1}}, \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}})$. Then $\frac{p(Z)}{2\sqrt{S(Z)}} \leq \max(\sqrt{\frac{\mu}{\mu-1}}, \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}})$. □

If $\mu < 3$, then $\max(\sqrt{\frac{\mu}{\mu-1}}, \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}}) = \sqrt{\frac{\mu}{\mu-1}}$. With such a value of μ, the dominant worst case is when the rectangle R is a perfect square (see Figure 9(a)) and the low value of μ implies the possibility for the square to take a large portion of the rectangle. If $\mu > 3$, then $\max(\sqrt{\frac{\mu}{\mu-1}}, \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}}) = \frac{\sqrt{\mu(\mu+1)}}{2\sqrt{\mu-1}}$. With such a value of μ, the dominant worst case is when the rectangle R has an aspect ratio of μ (see Figure 9(b)) and the large value of μ implies a huge deformation of the rectangle what increases the approximation ratio.

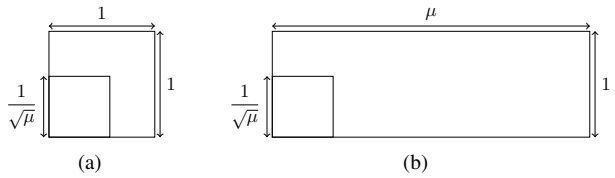

Figure 9: Dominant worst case depending on μ for the Square operation.

In summary, $x \mapsto \sqrt{\frac{x}{x-1}}$ is decreasing on $]1,3]$ and $x \mapsto \frac{\sqrt{x(x+1)}}{2\sqrt{x-1}}$ is increasing on $[3, +\infty[$, therefore, with this proof technique and without a change in $SNRRP$, the value

$\mu = 3$ yields the best possible approximation ratio. It is thus necessary to include additional cases to improve over $SNRRP$.

3) Description of NRRP: Intuitively, choosing a larger value of μ appears to be counter productive (since it tends to produce taller and skinnier rectangles), and we thus work with $\mu = \frac{5}{2}$, which is the lowest value of μ such that $p = 2$ in Lemma 9, allowing to use the Tripartition routine. However, as explained above, achieving an improved approximation ratio requires to design more sophisticated recursive partitionings to avoid the worst case which happens when the remaining large value is not large enough. The actual $NRRP$ generates 10 sub-cases (see Figure 10), all of which are fully detailed in [5]. The complete depiction of NRRP and of its subroutines is given in the supplementary material.

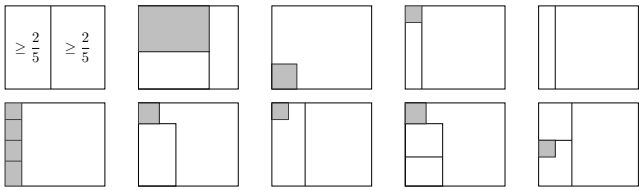

Figure 10: All the cases in NRRP. In each case, the gray rectangle is the one on which the recursive call is made, the white ones are returned zones.

Theorem 12 (from [5])**.** *NRRP is a $\frac{2}{\sqrt{3}}$-approximation for PERI-SUM.*

V. Comparison of NRRP and Optimal

In this section, we compare the results returned by NRRP and the known optimal partitioning for a large number of computational power ratios (CP ratios), under two restrictions. First, we are concerned with the sum of half perimeters (SHP) only, which for convex rectilinear partitions is equivalent to the sum of the projections of partition areas along the axes, as described in Section II. Second, we compare NRRP to the optimal partitionings for two and three partition arrangements only. The first restriction is justified for several reasons. First, both NRRP and the known optimal partitionings deliver perfect load balance, leaving communication the only parameter to be optimized. Second, the SHP is the most simple yet useful metric for comparing the communication volume of two load-balanced partitionings. Finally, SHP can be used for any number of partitionings and thus is not limited to our second restriction. The second restriction is justified since it is only for two and three partition arrangements that the optimal partitionings are known for all CP ratios.

In order to carry out this comparison we implemented simulations that compute the SHP of partitionings returned by NRRP and the SHP of the corresponding optimal partitioning for a large number of CP ratios across all ratio ranges possible. This is done for the two and three partition cases separately. Due to the complexity of the highly-recursive NRRP algorithm, two simulation implementations were employed. The two implementations were written in Python and Java by separate team members who did not communicate until results were complete, after which results were compared, and any inconsistencies resolved.

A. Two Partitions

For two partitions there are only two optimal partitionings: the Straight Line partitioning, achieved with one call of the *Guillotine* subroutine of NRRP described in Section IV-B, and the Square Corner partitioning, achieved with one call of the *Square* subroutine of NRRP, also described in section IV-B. For CP ratios $s_Q : s_P$ where $s_Q \leq s_P$, normalized so that $s_Q + s_P = 1$, the Square Corner partitoning is optimal when $s_Q \leq 0.25$. The simulation analyzed CP ratios ranging from the only homogeneous case of $s_Q = s_P = 0.5$, to the most heterogeneous case of $0.005 : 0.995$ in steps of 5.0×10^{-3}. As NRRP can achieve both optimal partitionings in one step, calling subroutines that return known optimal shapes, it is not surprising that our simulations did not find any CP ratios where NRRP was not optimal.

B. Three Partitions

The three partition case is more complicated than the two partition case. There are three optimal partitionings as determined in Section III-B, but determining which one is optimal for a given ratio is not as straightforward as in the two processor case. With the same notations as in Section III, we denote the processors P, Q, R from the fastest to the slowest, and by s_P, s_Q, s_R their respective speeds. Figure 11 shows the possible values of s_R (on the x axis) and s_Q (on the y axis) for CP ratios $s_R : s_Q : s_P$ where $s_R \leq s_Q \leq s_P$, normalized so that $s_R + s_Q + s_P = 1$. This space is dominated by a large region where the Block Rectangle partitioning is optimal, flanked by two regions along the $y-$axis ($s_R <\approx 0.086$) where the Square Corner partitioning is optimal for values of $s_Q \leq 0.25$, and the Square Rectangle partitioning is optimal for values of $s_Q \geq 0.25$. At $s_Q = 0.25$ both are optimal provided $s_R <\approx 0.018$. These s_R values are presented as approximate but the exact values are known; they are the solutions of the quadratic expressions that define the various regions.

For three partitions, our simulation analyzed 867 CP ratios ranging from the only homogeneous case of $s_Q = s_R = s_P = 0.333333$ to the most heterogeneous case of $0.003333 : 0.003333 : 0.993333$. We started with the homogeneous case and incremented s_P by 0.01 until we reached $s_P = 0.993333$. For each s_P, we started with $\frac{1-s_P}{2} : \frac{1-s_P}{2} : s_P$ and then incremented s_Q by 0.01 and decremented s_R by 0.01, while $s_Q < s_P$ (and $s_R > 0$). This resulted in different numbers of CP ratios for each of the 67 values of s_P, totaling different 867 CP ratios.

For each CP ratio, if the difference between the SHP returned by NRRP and the optimal SHP was greater than 2.0×10^{-6}, then NRRP is considered to be non-optimal. We chose this value as all ratios were calculated to a precision of 1.0×10^{-6}. Under these conditions NRRP is optimal for a total of 596 CP ratios (just over 2/3 of all CP ratios). Although non-optimal for 276 CP ratios, NRRP is never far from optimal

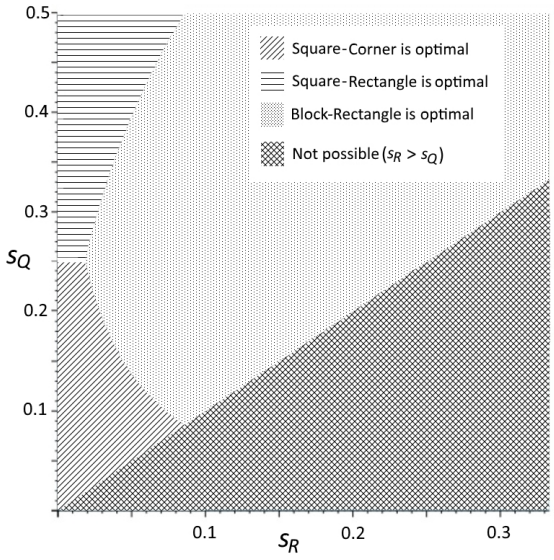

Figure 11: Regions where Square Corner, Square Rectangle, and Block Rectangle partitionings are optimal for three partitions, with $x = s_R$ and $y = s_Q$. Note that as $0 < s_R \leq s_Q \leq s_P$ and $s_R + s_Q + s_P = 1, s_R \leq \frac{1}{3}$ and $s_Q < \frac{1}{2}$.

(average of 1.94%, maximum of 7.49% above optimal). This maximum is below the bound of $\frac{2}{\sqrt{3}}$ established by Theorem 18 by a factor of over 2.

The y-axis of Figure 12 shows the percent of all CP ratios where NRRP is not optimal ($n = 867$), and the x-axis shows how much above optimal these CP ratios are, from 0 to 7.5%, in intervals of 0.5%. The height of each bar indicates the total percent of all CP ratios where NRRP is not optimal for each interval. The sub-bars indicate the percent of CP ratios where each partitioning shape is optimal for each interval. For instance, the left-most bar in Figure 12 shows that just over 10% of all CP ratios where NRRP is not optimal are less than 0.5% above optimal. Of this, Square Rectangle is optimal for just under 8%, Square Corner is optimal for just over 1%, and Block Rectangle is optimal for less than 1%. We discuss the case of each shape in detail below.

- NRRP is not optimal for 69 CP ratios where Square Rectangle is the optimal shape. Square Rectangle is dominant where NRRP is very nearly optimal – within 0.5% (x-axis, Figure 12). Our simulations showed that these are CP ratios where s_P is in the lower portion of its range, s_Q is in the upper half of its range, and s_R is extremely small ($s_P \leq 0.733333, s_Q \geq 0.253333, s_R \leq 0.028333$).
- NRRP is not optimal for 112 CP ratios where Square Corner is the optimal shape. Square Corner is dominant when NRRP is between 0.5% and 3.0% from optimal (x-axis, Figure 12). Our simulations showed that these are CP ratios where s_P is above its Square Rectangle range, s_Q is in the lower half of its range, and s_R is small ($s_P \geq 0.743333, s_Q \leq 0.248333, s_R \leq 0.083333$).
- NRRP is not optimal for 95 CP ratios where Block Rectangle is the optimal shape. Block Rectangle is dominant when NRRP is between 3.0% and 7.5% from optimal (x-axis, Figure 12). Our simulations showed that these are CP ratios where s_P is in the middle of its range, s_Q varies considerably, and s_R is small ($0.453333 \leq s_P \leq 0.823333, 0.088333 \leq s_Q \leq 0.458333, s_R \leq 0.123333$).

Although there is some overlap in CP ratio ranges where the three optimal shapes are dominant, some conclusions can be drawn. For instance, NRRP is always optimal when $s_R > 0.123333$. Additionally, the values of s_Q and s_P determine when Square Corner or Square Rectangle are optimal. Square Rectangle is only a possible optimal shape when $s_P <\approx 0.74$ and $s_Q >\approx 0.25$. Square-Corner is only a possible optimal shape when $s_P >\approx 0.74$ and $s_Q <\approx 0.25$. This 'y-boundary' is also clearly seen in Figure 11, which shows the optimal shape for all possible ratios, regardless of whether NRRP is optimal or not.

C. More Than 3 Partitions

We do not investigate cases of 4 (or more) partitions as it is not known how many optimal partitioning shapes there are, what they are, and for what ratios they are optimal. In our experience the three partition case is much more complex than the two partition case. We have no reason to believe that the four partition case will be any different, and thus it is beyond the scope of this work. Additionally, it is difficult to speculate as to how NRRP will perform in this case, but it is possible that NRRP performance decreases with increasing partitioning complexity as for two partitions we saw that NRRP is optimal for all ratios, but for three partitions we saw that NRRP was not optimal for approximately $\frac{1}{3}$ of ratios. Nonetheless, all NRRP partitionings will be within $\frac{2}{\sqrt{3}}$ of the optimal. This is one of the most important strengths of the NRRP algorithm - even where the optimal partitionings are not known (which currently is all partitionings with more than three partitions), NRRP returns a partitioning which is no more than a factor of $\frac{2}{\sqrt{3}}$ of the optimal.

However, in [5], some simulations with numbers of processors greater than 3 has been performed (up to 64 processors). In this set of experiments, three kinds of processors have been considered, each with different speed (one representing CPU, another GPU and the last accelerators) and for each value of n (number of processors), different repartitions of CPUs/GPUs/accelerators have been tested. As stated previously, for such numbers of processors, optimal solutions are not known and the comparison relies on the lower bound in Equation 1. With such settings, NRRP returns, on average, solutions whose SHP is within 5% of the lower bound. The maximum ratio between a solution returned by NRRP and the bound is 1.106. Note that solutions of RRP (Algorithm 1) have also been considered, with behavior similar to the one of NRRP, except that the worst case scenario is significantly worse (around 1.3).

VI. Conclusion

The problem of partitioning a matrix into a set of submatrices to address simultaneously the problems of load balancing and communication minimization for data parallel applications on heterogeneous platforms has been proved to be NP-Complete. In this paper, we present recent approaches

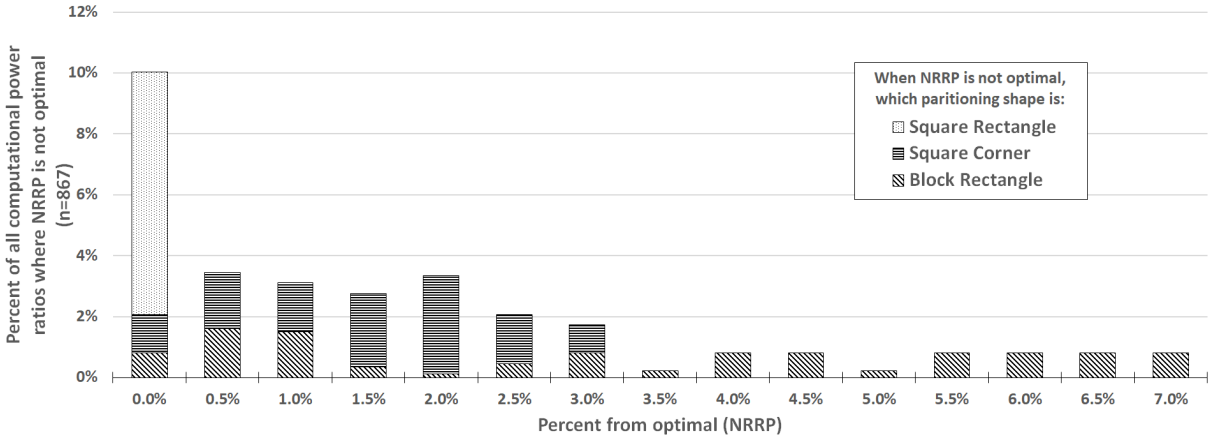

Figure 12: Percent from optimal for all computational power ratios where NRRP is not optimal, and which partition shape is optimal in these cases.

that relax the restriction that all partitions should be shaped as rectangles. The first approach uses an original mathematical technique to find the exact optimal partitioning for a small number of processors. This is important as in our experience there is an appetite for exact solutions, even in this restricted setting, particularly as it is applicable to current CPU/GPU/accelerator configurations. Due to the complexity of this technique, optimal algorithms are known for up to three heterogeneous resources only. However, even at this small scale, we can observe that optimal partitions are often non-rectangular and sometimes non-intuitive. This justifies the extension of the non-rectangular partitioning problem to find approximate solutions for any number of resources.

In this direction, we propose two algorithms, i.e. Simple Non-Rectangular Recursive Partitioning (SNRPP), which is a $\sqrt{\frac{3}{2}}$ approximation algorithm, and Non-Rectangular Recursive Partitioning (NRPP), which is a $\frac{2}{\sqrt{3}}$ approximation algorithm. While sub-optimal, these algorithms return results for arbitrary numbers of partitions, an advantage for problems with larger number of processors where the shape of optimal solutions is unknown. We use the first exact approach to analyze how close to the known optimal solutions the NRRP algorithm is for small numbers of partitions. We show that this approximation algorithm produces partitions that are often very close to optimal. Our improved algorithms have the potential for significant practical impact: in many applications, matrix multiplication operations are used iteratively a large number of times, so that even small improvements have a significant impact. We have observed this impact in our work, where a small difference in theory leads to a large difference in practice, particularly when problem size and network topology are taken into account. In one example, at a computational power ratio where the optimal partition shape outperforms the NRRP partition by only 1.86%, the optimal shape outperforms the NRRP partition by 23.63%. Thus, a nearly optimal solution in terms of communication cost can turn out to be far from the optimal in terms of real-world execution time.

This work opens several directions of future research. As far as optimal partitionings are concerned, it would be interesting to study cases with a larger number of resources (4, 5, ...) and to study the actual complexity of finding an optimal partition for the general case. Improvements of the approximation ratio might also be possible, for example by considering smaller values for the parameter μ. Another direction for future research would be to study other related models, in particular to consider the case of sparse rather than dense matrices, or to consider more precise communication models. In the sparse matrices case, the processing cost for a processor is related not to the area assigned to it, but to the number of non-zero elements in its zone, which introduces a new source of heterogeneity related to input matrices. More precise communication models would require network topology to be taken into account explicitly, as proposed in [4]. In this case, similar to those studied here, non-intuitive shapes are more dominant as heterogeneity increases, which provides a strong motivation to continue working on finding optimal solutions, even in the case of a small number of heterogeneous resources, but with explicit topologies.

Finally, the natural next step is to provide practical implementations of these algorithms in actual matrix multiplication kernels. This work has already been started [26], with work in two directions. First, this requires a move from continuous partitionings to the discrete partitioning of a $[1, N] \times [1, N]$ matrix. To the best of our knowledge, finding approximation algorithms for the discrete problem has not been addressed yet. Second, practical implementations also require dynamic work-stealing strategies to correct for the unpredictability of computing times inherent to large-scale machines.

VII. Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474. This work is partially supported by EU under the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS).

References

[1] B. A. Becker and A. Lastovetsky, "Matrix multiplication on two interconnected processors," in *2006 IEEE International Conference on Cluster Computing*. IEEE, 2006, pp. 1–9.

[2] A. DeFlumere, A. Lastovetsky, and B. A. Becker, "Partitioning for parallel matrix-matrix multiplication with heterogeneous processors: The optimal solution," in *International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2012, pp. 125–139.

[3] B. A. Becker and A. Lastovetsky, "Towards data partitioning for parallel computing on three interconnected clusters," in *ISPDC'07*. IEEE, 2007, pp. 39–39.

[4] A. DeFlumere, "Optimal partitioning for parallel matrix computation on a small number of abstract heterogeneous processors," PhD, University College Dublin, Dublin, 09/2014 2014.

[5] O. Beaumont, L. Eyraud-Dubois, and T. Lambert, "A new approximation algorithm for matrix partitioning in presence of strongly heterogeneous processors," in *IPDPS*, 2016.

[6] H. Nagamochi and Y. Abe, "An approximation algorithm for dissecting a rectangle into rectangles with specified areas," *Discrete applied mathematics*, vol. 155, no. 4, pp. 523–537, 2007.

[7] A. Fügenschuh, K. Junosza-Szaniawski, and Z. Lonc, "Exact and approximation algorithms for a soft rectangle packing problem," *Optimization*, vol. 63, no. 11, pp. 1637–1663, 2014.

[8] A. Kalinov and A. Lastovetsky, "Heterogeneous distribution of computations solving linear algebra problems on networks of heterogeneous computers," *Journal of Parallel and Distributed Computing*, vol. 61, no. 4, pp. 520–535, 2001.

[9] O. Beaumont, V. Boudet, F. Rastello, Y. Robert *et al.*, "Partitioning a square into rectangles: NP-completeness and approximation algorithms," *Algorithmica*, vol. 34, no. 3, pp. 217–239, 2002.

[10] B. A. Becker, "High-level data partitioning for parallel computing on heterogeneous hierarchical computational platforms," Ph.D. dissertation, University College Dublin, 2010.

[11] B. A. Becker and A. Lastovetsky, "Max-plus algebra and discrete event simulation on parallel hierarchical heterogeneous platforms," in *Euro-Par 2010 Parallel Processing Workshops*. Springer Berlin Heidelberg, 2011, pp. 63–70.

[12] A. DeFlumere and A. Lastovetsky, "Searching for the optimal data partitioning shape for parallel matrix matrix multiplication on 3 heterogeneous processors," in *IPDPSW*. IEEE, 2014, pp. 17–28.

[13] ——, "Optimal data partitioning shape for matrix multiplication on three fully connected heterogeneous processors," in *European Conference on Parallel Processing*. Springer, 2014, pp. 201–214.

[14] O. Beaumont, A. Legrand, F. Rastello, and Y. Robert, "Static LU decomposition on heterogeneous platforms," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, 2001.

[15] O. Beaumont, V. Boudet, A. Petitet, F. Rastello, and Y. Robert, "A proposal for a heterogeneous cluster ScaLAPACK (dense linear solvers)," *IEEE Transactions on Computers*, vol. 50, no. 10, pp. 1052–1070, 2001.

[16] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet *et al.*, *ScaLAPACK users' guide*. Siam, 1997, vol. 4.

[17] D. Clarke, A. Ilic, A. Lastovetsky, and L. Sousa, "Hierarchical partitioning algorithm for scientific computing on highly heterogeneous CPU + GPU clusters," in *European Conference on Parallel Processing*. Springer, 2012, pp. 489–501.

[18] R. Shams and P. Sadeghi, "On optimization of finite-difference time-domain (FDTD) computation on heterogeneous and GPU clusters," *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 584–593, 2011.

[19] N. Mohamed, J. Al-Jaroodi, and H. Jiang, "DDOps: dual-direction operations for load balancing on non-dedicated heterogeneous distributed systems," *Cluster Computing*, vol. 17, no. 2, pp. 503–528, 2014.

[20] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "StarPU: a unified platform for task scheduling on heterogeneous multicore architectures," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 2, pp. 187–198, 2011.

[21] J. Planas, R. M. Badia, E. Ayguadé, and J. Labarta, "Hierarchical task-based programming with StarSs," *International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 284–299, 2009.

[22] A. YarKhan, J. Kurzak, and J. Dongarra, "QUARK users' guide: Queueing and runtime for kernels," *University of Tennessee Innovative Computing Laboratory Technical Report ICL-UT-11-02*, 2011.

[23] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Hérault, and J. J. Dongarra, "PaRSEC: Exploiting heterogeneity to enhance scalability," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 36–45, 2013.

[24] O. Beaumont, L. Eyraud-Dubois, A. Guermouche, and T. Lambert, "Comparison of static and dynamic resource allocation strategies for matrix multiplication," in *26th IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2015*, 2015.

[25] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Minimizing communication in linear algebra," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 3, pp. 866–901, Jul. 2011, arXiv: 0905.2485. [Online]. Available: http://arxiv.org/abs/0905.2485

[26] T. Lambert, "On the Effect of Replication of Input Files on the Efficiency and the Robustness of a Set of Computations," Ph.D. dissertation.

Olivier Beaumont received a PhD degree in Computer Science from University of Rennes (1999) and its Habilitation à diriger des Recherches from the University of Bordeaux (2004). His main research interests include scheduling and load balancing problems in the context of High Performance Computing. He is the author of more than one hundred research papers and has acted as Algorithm Track Chair for major HPC conferences. Since 2007, he is a senior researcher at Inria Bordeaux Sud-Ouest in the Realopt team.

Brett A. Becker received a PhD degree in computer science from University College Dublin, a MA degree in higher education from the Dublin Institute of Technology, and a MSc degree in computational science from University College Dublin. His main research interests include heterogeneous parallel computing and computer science education. He has published over 20 papers in refereed journals and international conferences, and he is currently an assistant professor at UCD in Dublin, Ireland.

Ashley DeFlumere received a PhD degree in computer science from University College Dublin. Her main research interests include parallel and scientific computing, mathematical modeling, and computer science education. She is currently a lecturer at Wellesley College, USA.

Lionel Eyraud-Dubois received a PhD degree in Computer Science from Université de Grenoble. His main research interests include combinatorial optimization and approximation algorithms, in particular for scheduling and resource allocation problems in computer systems. He is currently a full-time researcher at Inria Bordeaux Sud-Ouest in the Realopt team, and a member of the Université de Bordeaux.

Thomas Lambert received a master degree from ENS of Lyon, France, and a PhD degree from the University of Bordeaux, France, in 2017. He is currently a research associate in the University of Manchester, UK. His main interests in research are scheduling aspects of problems linked to High-Performance Computing and Big Data applications, in particular issues related to replication of data in parallel computing.

Alexey Lastovetsky received a PhD degree from the Moscow Aviation Institute and a Doctor of Science (Habilitation) degree from the Russian Academy of Sciences. His main research interests include algorithms, models, and programming tools for high performance heterogeneous computing. He published over a hundred technical papers in refereed journals, edited books, and international conferences. He authored the monographs Parallel computing on heterogeneous networks (Wiley, 2003) and High performance heterogeneous computing (Wiley, 2009).