On the Complexity of the Generalized Block Distribution

Michelangelo Grigni[1] and Fredrik Manne[2]

[1] Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322, USA
[2] Department of Informatics, University of Bergen, N-5020 Bergen, Norway

Abstract. We consider the problem of mapping an array onto a mesh of processors in such a way that locality is preserved. When the computational work associated with the array is distributed in an unstructured way the generalized block distribution has been recognized as an efficient way of achieving an even load balance while at the same time imposing a simple communication pattern.
In this paper we consider the problem of computing an optimal generalized block distribution. We show that this problem is NP-complete even for very simple cost functions. We also classify a number of variants of the general problem.
Keywords: Load balancing, parallel data structures, scheduling and mapping

1 Introduction

A basic task in parallel computing is the partitioning and subsequent distribution of data among processors. The problem one faces in this operation is how to balance two often contradictory aims; finding an equal distribution of the computational work and at the same time minimizing the imposed communication.
For data stored in an array several high performance computing languages allow the user to specify a partitioning and distribution of data onto a logical set of processors. The compiler then maps the data onto the physical processors and determines the communication pattern. An example of such a scheme is the block distribution found in languages such as Vienna Fortran [1] and HPF [7]. This mapping results in equal size blocks and therefore cannot adapt to any load imbalance which might be present.
More general partitioning schemes which have been proposed for these kinds of problems include the generalized and semi-generalized block distribution [2, 12, 13, 14]. The generalized block distribution preserves the array-structured communication of the block distribution while at the same time allowing for different sized blocks.
In [10] a number of algorithms were described for computing a well-balanced generalized block distribution. These were compared with other distribution schemes which showed that in many cases the generalized block distribution can give a good load balance while at the same time maintaining a simple communication pattern.

In this paper we show that the problem of computing a generalized block distribution of cost less than some constant K is NP-complete, even for very simple cost functions. This implies that one cannot generally expect to compute an optimal generalized block distribution.
The outline of this paper is as follows: In Section 2 we give a formal definition of the problem, in Section 3 we show that the problem of determining whether there exists a solution of cost less than K is NP-complete, and finally in Section 4 we discuss variants of this problem and point to some open problems.

2 The Generalized Block Distribution

For integers a and b, let $[a,b]$ denote the interval of integers $\{a, a+1, \ldots, b\}$ (empty if $a > b$). Let $[a]$ denote $[1,a]$.
Given $A \in \Re^{m \times n}$ and integers p and q such that $p \in [m]$ and $q \in [n]$. Let $R = (r_0, r_1, \ldots, r_p)$ be a sequence of integers such that $1 = r_0 \leq r_1 \leq \ldots \leq r_p = m+1$. Then R defines a *partition* of $[m]$ into the p intervals $[r_i, r_{i+1} - 1]$, for i in $[0, p-1]$. We denote each interval by R_i. Note that some intervals may be the empty interval.

Definition 1 General Block Distribution. Given A, p and q as above, a *generalized block distribution* consists of a partition of $[m]$ into p intervals and of $[n]$ into q intervals, so that A partitions into $p \times q$ contiguous blocks. For $i \in [p]$ and $j \in [q]$, we denote the ijth block by A_{ij}.

See Fig. 1 for an example of the generalized block distribution.
The generalized block distribution was first discussed by Fox *et al.* [4] and implemented as part of Superb environment [14] and later in Vienna Fortran [3]. It is also a candidate to be included as part of the ongoing HPF2 effort [8]. See [2] and [9] for examples of how the generalized block distribution can be used in areas such as sparse-matrix and particle-in-cell computations.
In a parallel environment the time spent on a computation is determined by the processor taking the longest time. To estimate the time needed to process each block we define a non-negative cost function ϕ on contiguous blocks of A. We assume that if a and b are blocks of A such that a is contained in b then $\phi(a) \leq \phi(b)$, and that $\phi(a) = 0$ if and only if a is the empty block. For reasonable functions ϕ we also expect that if the value of $\phi(a)$ (or $\phi(b)$) is known then the value of $\phi(b)$ (or $\phi(a)$) can be computed in $O(|b| - |a|)$ time. An example of ϕ might be the number of non-zero elements, or the sum of the absolute values of the elements in a block.
Then the natural optimization problem is to find a generalized block distribution that minimizes the maximum ϕ over all blocks. The equivalent decision problem is the following:

> **(GBD)** Instance: A, p, q and ϕ as above, and an integer K.
> Question: Does there exist a generalized block distribution on A such that
>
> $$\max_{i \in [p], j \in [q]} \phi(A_{ij}) \leq K \ .$$

8	1	3	7	1	1
3	3	1	2	0	1
1	2	0	0	1	1
2	4	2	3	1	2
3	1	0	3	1	2
6	6	2	2	3	1

Fig. 1. An example of the generalized block distribution

3 GBD is NP-Complete

In this section we show that GBD is NP-complete. First we note that a solution to GBD can be verified efficiently, so GBD is in NP. To show that GBD is NP-complete we will reduce the NP-complete problem "Balanced Complete Bipartite Subgraph" [5, GT24] to GBD.

> [Problem GT24] Balanced Complete Bipartite Subgraph **(BCBS)**
> Instance: Given a bipartite graph $G = (V_1, V_2, E)$, and a positive integer K.
> Question: Are there subsets $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ such that $|U_1| = |U_2| = K$, and such that $u \in U_1$ and $v \in U_2$ imply $(u, v) \in E$ (that is, $U_1 \times U_2 \subseteq E$)?

Note that we may add isolated vertices above to assure that $|V_1| = |V_2|$. We now transform BCBS to a problem on the bipartite complement of G. That is, graph G' has the same vertex sets and the edge set $E' = V_1 \times V_2 - E$. Also let $K' = |V_1| - K$. We now have a problem equivalent to BCBS:

> Balanced Bipartite Cover **(BBC)**
> Instance: Given a bipartite graph $G' = (V_1, V_2, E')$ with $|V_1| = |V_2|$ and a positive integer K'.
> Question: Are there subsets $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ such that $|U_1| = |U_2| = K'$, and such that each edge $(u, v) \in E'$ has either $u \in U_1$ or $v \in U_2$?

It is clear from the construction of G' that BBC is NP-complete. Note that a solution for BBC leaves no "uncovered" edges between $V_1 - U_1$ and $V_2 - U_2$. Thus BBC may be formulated in terms of the adjacency matrix of G':

> Is it possible to choose K rows and K columns of a matrix so that these rows and columns contain all the non-zero entries?

We now show how to reduce an instance (G', K') of BBC to a particular instance (A, p, q, K, ϕ) of GBD. In fact we will have $K = 1$ and ϕ equal to the number of non-zero elements in a block.

Let $n = |V_1| = |V_2|$ in the given instance of BBC. We construct a $2(n+1) \times 2(n+1)$ zero-one matrix A as part of the GBD instance. The rows of A are labeled (in order) $\{s_{0,0}, s_{0,1}, s_{1,0}, s_{1,1}, \ldots s_{n,0}, s_{n,1}\}$, and similarly the columns are labeled $\{t_{0,0}, t_{0,1}, t_{1,0}, t_{1,1}, \ldots t_{n,0}, t_{n,1}\}$. The following entries of A are set to one:

1. $(s_{0,0}, t_{0,0})$
2. $(s_{0,0}, t_{2i,1})$ and $(s_{0,0}, t_{2i+1,0})$ for $0 \leq i < \lceil n/2 \rceil$
3. $(s_{0,1}, t_{2i+1,1})$ and $(s_{0,1}, t_{2i+2,0})$ for $0 \leq i < \lfloor n/2 \rfloor$
4. $(s_{2i,1}, t_{0,0})$ and $(s_{2i+1,0}, t_{0,0})$ for $0 \leq i < \lceil m/2 \rceil$
5. $(s_{2i+1,1}, t_{0,1})$ and $(s_{2i+2,0}, t_{0,1})$ for $0 \leq i < \lfloor m/2 \rfloor$
6. $(s_{i,0}, t_{j,0})$ and $(s_{i,1}, t_{j,1})$, for all $(i,j) \in E'$.

All other entries of A are set to zero. The first two rows and columns of the matrix in Fig. 2 illustrate how rules 1 through 5 effect A. If we are to find a solution to GBD with $K = 1$, these elements force us to at least place $n+1$ horizontal and $n+1$ vertical delimiters as shown by the dotted lines. Setting $p = q = n + K' + 2$, this leaves us with K' horizontal delimiters and K' vertical delimiters to partition the remaining matrix. For each edge in G', rule 6 constructs a 2×2 block in A with ones on the diagonal as shown in Fig. 2. Each such block must be split by either a horizontal or a vertical line (or both) if we are to achieve a cost of at most 1. Splitting such a block with a horizontal delimiter corresponds to choosing a vertex from V_1 in BBC, and splitting it with a vertical delimiter corresponds to choosing a vertex in V_2. It is clear from the construction of this matrix that there exists a solution to this GBD problem if and only if the corresponding BBC problem has a solution. Thus we can state our main result:

Theorem 2. *GBD is NP-Complete.*

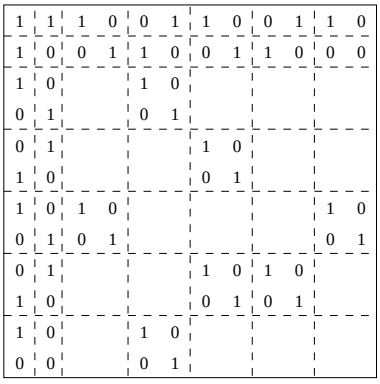

Fig. 2. Forcing the delimiters to create 2×2 squares

4 Conclusion

We have shown that GBD is NP-complete with ϕ equal to the number of elements in a block. This implies that GBD remains NP-complete for any derived cost function such as the sum of the elements in a block. Thus we have to settle for approximation algorithms to achieve an even load balance for this distribution. In a recent development [6] it has been shown that if $p = q$ with ϕ equal to the sum of the elements in a block then one of the algorithms in [10] gives a solution that is guaranteed to be within a bound of $4\sqrt{p}$ of the optimal.

We note that the following three variants of GBD can be solved in polynomial time:

- $n = q = 1$. The problem now becomes to partition a vector of length m into p segments. This problem has been studied extensively and the current fastest algorithm for computing an optimal solution runs in time $O(p(m-p))$ [11].
- p is fixed. Assume that we are given a fixed horizontal partition. The cost of a vertical interval is now defined to be the maximum cost of the p blocks inside this interval. Using this cost function this problem becomes equivalent to the one dimensional case. Since there are only polynomial many placements of the $p-1$ horizontal delimiters this problem is also solvable in polynomial time.
- If we relax how the partitioning is done in one dimension we get the semi-generalized block distribution where the interval $[m]$ is partitioned into p consecutive intervals R_i, $1 \leq i \leq p$ without restrictions on the size of $r_{i+1} - r_i$ and for each horizontal interval R_i, the interval $[n]$ is partitioned into q intervals. In [10] an algorithm is given to computes an optimal semi-generalized block distribution in time $O(pqm(m-p)(n-q))$.

If, instead of $|V_1| = |V_2| = K$, we have the restriction $|V_1| + |V_2| = K$ then the BCBS problem is in P [5], by reduction to matching. We note that the corresponding problem for the generalized block distribution is of no immediate practical interest. This is because the number of processors $p \times q$ usually is fixed. Thus the more relevant question is if given the number of processors r, is it possible to find a factorization of $r = p \times q$ that solves the GBD problem. However, as the following shows this problem still remains NP-complete. Given an instance of GBD with cost matrix A, $K = 1$, and ϕ equal to the number of elements in a block. Let g be the smallest prime such that $g > \max\{p, q\}$ and let $r = g^2$. We construct a new matrix C with A and a matrix B on the diagonal where B consists of a $(g-p) \times (g-q)$ block of all ones. In addition we set $c_{g,n} = c_{m,g} = 1$. All other elements of C are set to zero. Any solution to this problem requires that B is separated from A and that B is completely partitioned using $g-p$ horizontal delimiters and $g-q$ vertical ones. Since the only factorizations of r are $r = 1 \times g^2$, $r = g^2 \times 1$, and $r = g \times g$ it follows that r must be factored into $g \times g$ if we are to obtain a positive solution. This leaves $p-1$ horizontal delimiters and $q-1$ vertical ones to partition A. Thus this problem can be solved if and only if we can solve the corresponding GBD problem as well.

Another case of interested is the symmetric generalized block distribution. Here we assume that $m = n$ and $p = q$ and we add the extra restriction to any solution that $p_i = q_i$ for $1 \leq i \leq p$. This means that the diagonal blocks will be square and the diagonal elements of the matrix will lie on the diagonal processors. This is very convenient if one wants to gather a vector along the rows and then distribute the result along the columns. This is a typical situation in iterative linear solvers where one is performing series of matrix-vector multiplications. This problem appears simpler than the general problem since the number of possible solutions is reduces from $\binom{m}{p} \times \binom{n}{q}$ to $\binom{m}{p}$. However, we do not know the complexity of this problem.

References

1. B. Chapman, P. Mehrotra, and H. Zima, *Programming in Vienna Fortran*, Sci. Prog., 1 (1992), pp. 31–50.
2. ———, *High performance Fortran languages: Advanced applications and their implementation*, Future Generation Computer Systems, (1995), pp. 401–407.
3. ———, *Extending HPF for advanced data parallel applications*, IEEE Trans. Par. Dist. Syst., (Fall 1994), pp. 59–70.
4. G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors*, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1988.
5. M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, 1979.
6. M. Halldorsson and F. Manne. Private communications.
7. High Performance Fortran Forum, *High performance language specification. Version 1.0*, Sci. Prog., 1–2 (1993), pp. 1–170.
8. *High Performance Fortran Forum Home Page.* http://www.crpc.rice.edu/HPFF/home.html.
9. F. Manne, *Load Balancing in Parallel Sparse Matrix Computations*, PhD thesis, University of Bergen, Norway, 1993.
10. F. Manne and T. Sørevik, *Structured partitioning of arrays*, Tech. Rep. CS-96-119, Department of Informatics, University of Bergen, Norway, 1996.
11. B. Olstad and F. Manne, *Efficient partitioning of sequences*, IEEE Trans. Comput., 44 (1995), pp. 1322–1326.
12. M. Ujaldon, S. D. Sharma, J. Saltz, and E. Zapata, *Run-time techniques for parallelizing sparse matrix problems*, in Proceedings of 1995 Workshop on Irregular Problems, 1995.
13. M. Ujaldon, E. L. Zapata, B. M. Chapman, and H. P. Zima, *Vienna-Fortran/HPF extensions for sparse and irregular problems and their compilation.* Submitted to IEEE Trans. Par. Dist. Syst.
14. H. Zima, H. Bast, and M. Gerndt, *Superb: A tool for semi-automatic MIMD/SIMD parallelization*, Parallel Comput., (1986), pp. 1–18.

This article was processed using the LaTeX macro package with LLNCS style