Linear Scheduling is Nearly Optimal

Alain Darte, Leonid Khachiyan and Yves Robert*

Laboratoire LIP-IMAG, Ecole Normale Supérieure de Lyon
69364 LYON Cedex 07
e-mail: [darte,yrobert]@lip.ens-lyon.fr

Dept. of Computer Science, Rutgers University
New Brunswick, NJ, 08903
and Computing Center of the USSR Academy of Sciences, Moscow
e-mail: leonid@cs.rutgers.edu

November 1991

Abstract

This paper deals with the problem of finding optimal schedulings for uniform dependence algorithms. Given a convex domain, let T_f be the total time needed to execute all computations using the free (greedy) schedule and let T_l be the total time needed to execute all computations using the optimal linear schedule. Our main result is to bound T_l/T_f and $T_l - T_f$ for sufficiently "fat" domains.

Keywords Uniform dependence algorithms; Convex domain; Free schedule; Linear schedule; Optimal schedule; Path packing.

*Supported by the Project C3 of the French Council for Research CNRS, and by the ESPRIT Basic Research Action 3280 "NANA" of the European Economic Community. Part of this work has been done while the first author was visiting the CS Department at Rutgers University in October 1991.

1 Introduction

The pioneering work of Karp, Miller and Winograd [5] has considered a special class of algorithms characterized by *uniform data dependencies* and *unit-time computations*. This special class of algorithms, termed *uniform dependence algorithms* has proven of paramount importance in various fields of applications, such as systolic array design [2, 6, 8, 12, 13, 16, 17] and parallel compiler optimization [1, 3, 9, 10, 11, 18, 19, 20].

This paper deals with the problem of finding optimal schedulings for uniform dependence algorithms. We assume that such a schedule exists, and we refer to the seminal paper of Karp, Miller and Winograd [5] for necessary and sufficient conditions establishing the existence of a schedule over the nonnegative orthant.

Given a convex domain, let T_f be the total time needed to execute all computations using the *free (greedy) schedule* and let T_l be the total time needed to execute all computations using the *optimal linear schedule* (we define formally these schedules in the next section). Our main result is to bound T_l/T_f and $T_l - T_f$ for sufficiently "fat" domains, thereby extending results of Karp, Miller and Winograd. This is a useful result in practice, because linear schedules have been proposed for many scientific algorithms. Also, with linear schedules, code generation for the parallel execution of loop nests can be performed automatically and with a low overhead [10, 18]. See Fortes and Parisi-Presicce [4] for a further discussion upon the advantages of linear schedules.

The paper is organized as follows: first we define formally uniform dependence algorithms and schedulings. Next we review the results of Karp, Miller and Winograd [5], and those of Shang and Fortes [17], who have introduced an optimization method to determine the best linear schedule (section 2). Then we proceed to the proof of the main result (section 3). We give some conclusions in section 4.

2 Terminology and previous work

In this section we introduce some notations, and we summarize existing results, mostly due to Karp, Miller and Winograd [5].

2.1 Uniform Dependence Algorithms

Using the terminology of Shang and Fortes [17], a uniform dependence algorithm is defined as follows:

UDA: Uniform Dependence Algorithm A UDA can be described by an equation of the form:

$$v(j) = g_j(v(j - d_1), v(j - d_2), \ldots, v(j - d_m))$$

where

- **Domain of computation**: $j \in J \subset Z^n$ is an index point, J is the index set, and the positive integer n is the dimension of the algorithm. The index set J is described as a set of integer points (vectors) satisfying
$$J = \{x : Ax \leq b, A \in Z^{a \times n}, b \in Z^a, x \in Z^n\}$$
We write $J \equiv (A, b)$.

- **Unit-time computation**: g_j is the computation indexed by j, i.e. a single-valued function computed "at point j" in a single unit of time. v_j is the value computed at point j.

- **Dependence matrix**: $d_i \in Z^n, i = 1 \ldots m, m \geq 0$ are dependence vectors which are constant, i.e. independent of $j \in J$. The $n \times m$ matrix
$$D = \begin{pmatrix} d_1 & d_2 & \ldots & d_m \end{pmatrix}$$
is the dependence matrix of the algorithm.

In this paper, only structural information of the UDA, namely the index set $J \equiv (A, b)$ and the dependence matrix D is needed. Other information such as what computations occur at different points and where and when input/output of variables take place, can be ignored [1]. In the following, we let $Alg = (J, D)$ for defining a UDA. Unless specified otherwise, n is the dimension of index points, a is the number of constraints that define the shape of the domain, and m is the number of dependence vectors, so that the constraint matrix A is of dimension $a \times n$ and the dependence matrix D is of dimension $n \times m$.

[1] We point out that when referring to a point $j_2 = j_1 - d_i$ where $j_1 \in J$, $j_2 \not\in J$ for some dependence vector d_i, we assume that j_2 corresponds to an input data of the algorithm.

2.2 Scheduling a UDA

Given a UDA $Alg = (J, D)$, we write $j_1 \prec j_2$ when $j_2 = j_1 + d_i$ for some $d_i \in D$, i.e. when j_1 and j_2 are two points of J such that j_2 depends upon j_1.

Definition 1 *A **schedule** for a UDA $Alg = (J, D)$ is a function $\sigma : J \to Z$ such that for any arbitrary index points $j_1, j_2 \in J$, $\sigma(j_1) < \sigma(j_2)$ if $j_1 \prec j_2$.*

In other words, a schedule is a mapping which assigns a time of execution to each computation of the UDA in such a way that dependencies are preserved.

A *free schedule* schedules computations as soon as their operands are available. More formally:

Definition 2 *A schedule σ is called **free** if*

$$\sigma(j) = \begin{cases} 0 & \textit{if there is no } j' \in J \textit{ s.t. } j' \prec j \\ 1 + \max(\sigma(j'), j' \in J, j' \prec j) & \textit{otherwise} \end{cases}$$

Clearly, if there is a schedule, there is a unique free schedule. The free schedule, when it exists, is the "fastest" schedule possible. In the following, we always assume when considering a UDA that a schedule exists.

Consider a UDA $Alg = (J, D)$ for which a schedule exists, and let σ_{free} be the free schedule for the UDA. The total execution time is thus $T_f = 1 + \max(\sigma_{free}(j), j \in J)$.

Given a point $j \in J$, consider all dependence paths $(j_0, j_1, \ldots, j_g = j)$ that remain inside the index set J and terminate in j: we have $j_k \in J$ for $0 \le k \le g$ and $j_{k+1} = j_k + d_{i(k)}$, where $d_{i(k)} \in D$ for $0 \le k \le g - 1$. We can write $j - j_0 = Du$ where $u \in Z^n$ is a vector with nonnegative integer components. $\sigma_{free}(j)$ is equal to the maximum sum $u_1 + u_2 + \ldots + u_m$ that can be obtained in this way.

Next we introduce *linear schedules*, which have been proposed for the execution of many practical algorithms (see [17] and the references therein).

Definition 3 *For a UDA $Alg = (J, D)$, a **linear schedule** is a mapping $\sigma_\Pi : J \to Z$ such that*

$$\sigma_\Pi(j) = \lfloor \Pi j + c \rfloor \textit{ for } j \in J$$

where the linear schedule vector $\Pi \in Q^{1\times n}$ is such that $\Pi D \ge 1$ (which means $\Pi d_i \ge 1$ for all $d_i \in D$). The constant c is the offset: $c = -\min(\Pi j, j \in J)$.

The fact that σ_Π is indeed a schedule is due to the condition $\Pi D \ge 1$ which ensures that dependencies are preserved.

2.3 Free schedule versus piecewise linear schedule

Consider a UDA $Alg = (J, D)$ for which a schedule exists, and let σ_{free} be the free schedule for the UDA. Karp, Miller and Winograd [5] have considered UDAs with index set

$$F_n = \{j \in Z^n / j_i \geq 1,\ i = 1, \ldots, n\}$$

They have shown how to link the existence of a schedule for all points in F_n with the solution of the following two dual problems:

$$I \begin{cases} j - \sum_{i=1}^{m} u_i d_i \geq 0 \\ u_i \geq 0, i = 1, 2, \cdots, m \\ \max \sum_{i=1}^{m} u_i \end{cases} \qquad II \begin{cases} d_i.x \geq 1, i = 1, 2, \cdots, m \\ x_i \geq 0, i = 1, 2, \cdots, n \\ \min j.x \end{cases}$$

A schedule exists iff for every $j \in F_n$ the two dual problems have a common solution, denoted $m(j)$. Clearly, all dependence paths that remain in F_n and terminate in j lead to a feasible solution to problem I, hence $\sigma_{free}(j) \leq m(j)$. Karp, Miller and Winograd prove, among other results, that there exists a constant K such that $m(j) - \sigma_{free}(j) \leq K$ for all those points $j \in F_n$ that are not "too close" to the domain boundary.

Problem II can be interpreted in terms of finding for each point $j \in F_n$ the optimal linear schedule vector $x(j)$ that will lead to execution of point j at time $m(j)$. Such a vector $x(j)$ is necessarily an extremal point of the domain

$$\{x \in Q^n,\ d_i.x \geq 1 \, \text{for} \, 1 \leq i \leq m,\ x_i \geq 0 \, \text{for} \, 1 \leq i \leq n\}$$

Hence there is a finite number of such vectors and we see that this strategy leads to piecewise linear schedules, the same vector x being the optimal linear vector over a whole subregion of F_n.

2.4 Optimal (global) linear schedule

Consider again a UDA $Alg = (J, D)$ for which a schedule exists, and let σ_Π be a linear schedule. The total execution time will be

$$\begin{aligned} T_\Pi &= 1 + \max(\sigma_\Pi(j),\, j \in J) \\ &= 1 + \max(\lfloor \Pi.j_2 \rfloor - \lfloor \Pi.j_1 \rfloor,\ j_1, j_2 \in J) \end{aligned}$$

The best linear schedule is the one that minimizes T_Π over all rational vectors Π such that $\Pi D \geq 1$. We write

$$T_l = \min(T_\Pi,\ \Pi \in Q^n,\ \Pi D \geq 1) \quad (*)$$

The problem of determining a linear schedule vector Π achieving execution time in T_l has been considered by Shang and Fortes [17]. Their approach is to partition the solution space of all possible candidate vectors Π into convex subcones, and to solve a linear fractional problem for each of these subcones, in order to determine at compile time a subset of vectors containing the optimal solution.

At this point a natural question arises. Karp, Miller and Winograd have given a bound that links the value of the free schedule to the value of a linear schedule local to each point of the computation domain J. From a global (or macroscopic) point of view, can we establish any relation between T_f and T_l, the *total* execution time achieved with the free schedule and the optimal linear schedule respectively ? We call this point of view *macroscopic* because we do not mind if a given point is executed very late with the optimal linear schedule as compared to the free schedule. What really matters for the parallel execution time is the difference $T_l - T_f$.

An experimental answer has been given by Fortes and Parisi-Presicce [4]. They have computed the difference $\delta = T_l - T_f$ for 25 UDAs, whose domain was a n-parallelepiped, with $2 \leq n \leq 4$. They report the value $\delta = 0$ for 23 algorithms out of the 25, and $\delta = 1$ for the last two algorithms. They also point out that δ remains invariant with changes in the size of the index set of the algorithms.

The main result of this paper is to prove that the difference $T_l - T_f$ is indeed bounded by a constant K for any UDA $Alg = (J, D)$ with $J \equiv (A, b)$ sufficiently "fat". The constant K depends upon the shape of the domain (constraint matrix A) and upon the dependence vectors (dependence matrix D).

3 Comparing T_l and T_f

3.1 Dual problems

Consider the usual continuous relaxation of the integer programming problem of $(*)$. The associated dual problem will give us an interpretation of the parallel execution time as the length of a dependence path.

The time T_l of the best linear schedule for computing the domain is less than $2 + \lfloor T_l^* \rfloor$, where T_l^* is the solution of:

Problem I:

$$\min_{(XD \geq 1)} \max_{((p,q) \in (\Omega, \Omega))} X(p - q)$$

where

$$\Omega = \{x : Ax \leq b, A \in Z^{a \times n}, b \in Z^a, x \in \mathcal{R}^n\}$$

According to Von Neumann's saddlepoint theorem [14, p. 393], this value is equal to:

$$\max_{((p,q) \in (\Omega,\Omega))} \min_{(XD \geq 1)} X(p-q)$$

Now, consider the value $\min_{(XD\geq 1)} X(p-q)$ as the solution of the linear problem:

$$\begin{cases} XD \geq 1 \\ \min X(p-q) \end{cases}$$

By the duality theorem of linear programming, this value is the same as:

$$\begin{cases} Y \geq 0 \\ DY = p - q \\ \max \sum_{i=1}^{m} Y_i \end{cases}$$

Thus, $2 + \lfloor T_l^* \rfloor$ gives an upper bound on the execution time of the linear schedule X over J, where T_l^* is the solution of:

Problem II:

$$\begin{cases} y \geq 0 \\ p \in \Omega \\ q \in \Omega \\ Dy = p - q \\ \max \sum_{i=1}^{m} y_i \end{cases}$$

The interpretation of the value given by Problem II is the length of the longest path given as a linear combination of dependence vectors, a priori with rational components, whose both ends are in the domain Ω.

The problem now is that we want to find an actual path of dependences, which means a linear combination of dependence vectors, but with integer components, and for which all nodes, not only the first and the last one, are within the domain. This will allow us to give a lower bound on T_f, the time of the free schedule.

3.2 A problem of path packing

Suppose we are given $(m+1)$ vectors $d_1, \ldots, d_m, b$ such that

$$(1) \quad d_1 X_1 + \cdots + d_m X_m = b$$

where $X_1, \ldots, X_m$ are nonnegative integers, and let Ω be a given domain. We say that (1) can be **packed** in Ω, if there is a path from 0 to b that uses each of the vectors $d_1, \ldots, d_m$ exactly $X_1, \ldots, X_m$ times and stays within Ω. Consider the following problem: given a convex body $\mathcal{K}$ containing $0, d_1, \ldots, d_m$, give an upper bound on the homothetic coefficient λ such that (1) can be packed in Conv.Hull$\{(\lambda\mathcal{K}) \cup (b + \lambda\mathcal{K})\}$.

Lemma 1 *λ is less than $2m$.*

Proof: The proof is somewhat similar to the proof of lemma 3 in [5]. Let $l = \sum_{i=1}^{m} X_i$, and consider the points $p_j = D\lfloor jX/l \rfloor, j = 0, \ldots, l$ and $q_j = DjX/l, j = 0, \ldots, l$, where $X = (X_1, \ldots, X_m)^t$. Note that each q_j lies on the line segment $[0, b]$ and that $q_j - p_j = D\delta^{(j)}$ where $0 \leq \delta_i^{(j)} \leq 1$. Since $\mathcal{K}$ is convex, $D\delta/(\sum_{i=1}^{m} \delta_i)$ belongs to $\mathcal{K}$ and $q_j - p_j$ belongs to $(\sum_{i=1}^{m} \delta_i)\mathcal{K}$, and a fortiori to $m\mathcal{K}$. It is easy to see that one can always add at most $m - 1$ intermediate points to link p_j to $p_{(j+1)}$ if p_j and $p_{(j+1)}$ differ by more than one component. Again, these new points $r_{j,k}$, $1 \leq k \leq k(j)$, where $k(j) \leq m - 1$, are such that $r_{j,k} - p_j \in m\mathcal{K}$. Now, all the points p_j, $r_{j,k}$ are linked by dependence vectors: the desired path can be constructed through this set of points, all of which belong to the convex hull of two domains $2m\mathcal{K}$ placed at the origin and b respectively. □

On the other hand, let $\mathcal{K}$ be a simplex, given by $(m+1)$ affine independent vertices $v_j, 0 \leq j \leq m$, and let $d_j = v_j - c$ where c is the centroid of $\mathcal{K}$. Then $\sum_{i=1}^{m} d_i = 0$ and $\lambda = m$ for any packing. Thus, for arbitrary $\mathcal{K}$, λ can grow linearly with m.

3.3 Bounding T_l/T_f

In this section we would like to "parametrize" in some sense the size of the index domain of the UDA. Our goal is to analyze how the ratio T_l/T_f evolves as a function of the domain size.

For the special case where the domain $\{Ax \leq b\}$ is a n-cube of edge size N (i.e. with N^n points), we can choose N as the size parameter. For a general convex $Ax \leq b$, how can we find such a parameter ? We use the following notation: if $P = \{Ax \leq b, x \in \mathcal{R}^n\}$, then tP denotes the polyhedron $\{Ax \leq tb, x \in \mathcal{R}^n\}$.

Definition 4 *A domain P is **fat** if*

(1) $0 \in P$

(2) $0 + d_i \in P$, $\forall i, 1 \leq i \leq m$

(3) $0 + e_i \in P$, $\forall i, 1 \leq i \leq n$, *where the* e_i*'s are the canonical basis vectors.*

The first condition is just a technical condition which consists in shifting the domain so that it contains 0, without loss of generality. Condition (2) means that the domain contains at least one pattern of the dependence system, which is a reasonable condition for practical cases. Condition (3) implies that the domain is not too thin nor too skewed. In practice, computation domains are often squares, cubes, triangles, or they can have more complicated shapes but they do not look like needles.

Now, considering a fat domain $\{Ax \leq b\}$ and the UDA over the family of domains $\{Ax \leq Nb\}$ enables us to parametrize the domain size as a function of N. N is the parameter size.

Definition 5 *The* ***fatness*** *of* P *is* $t = \max\{\tau > 0; \frac{1}{\tau}P \text{ is fat}\}$ *if it exists.*

(Remark: for a fixed n, the fatness can be computed in polynomial time by Lenstra's algorithm, see [15].)

Let X be the optimal solution of problem II for a fat domain P. Note that there always exist a solution X such that only n out of the m components of X are nonzero. Therefore, we can assume without loss of generality that $m \leq n$. X represents a path from $q \in P$ to $p \in P$ of length l where all these quantities are a priori rational. We are going to round them in three steps.

- Step 1: Consider $Y = \lceil X \rceil$. Y can be seen as a actual path of dependence from $q \in P$ to $p' \in (1 + \frac{n}{t})P$ of length greater than l (condition 2).

- step 2: Lemma 1 shows than there exists a path given by Y packed in a domain covered by copies of $\frac{2n}{t}P$ and thus in $(1 + \frac{3n}{t})P$.

- step 3: We can shift this path by rounding q and p' (condition 3) to have an integer path in between two integer points in $(1 + \frac{3n+1}{t})P$

Thus, we have the following lower bound:

$$T_f((1 + \frac{3n+1}{t})P) \geq l \geq T_l^*(P)$$

This holds also for the domain λP for any λ.

$$T_f((1 + \frac{3n+1}{\lambda t})\lambda P) \geq l \geq T_l^*(\lambda P) = \lambda T_l^*(P)$$

Now take $\lambda = 1 - (3n+1)/t$ and we obtain: $\frac{T_f}{T_l^*} \geq 1 - \frac{3n+1}{t}$. This is true for domains sufficiently fat, i.e. with fatness greater than $3n+1$. However, a similar bound for arbitrary domains does not exist, see [5] example 2.

Theorem 1 *For a UDA algorithm on a domain J of dimension n and with fatness t greater than $3n+1$, $1 \geq T_f/T_l^* \geq 1 - O(n/t)$ where $T_l^* \geq T_l - 1$. The relative error between the optimal linear schedule and the free schedule is bounded by $O(n/t)$.*

Now, consider the UDA over the family of domains $\{Ax \leq Nb\}$ where $\mathcal{D} = \{Ax \leq b\}$ has fatness 1. We have $\frac{T_f}{T_l^*} \geq 1 - \frac{3n+1}{N}$ and $T_l^*(N\mathcal{D}) = NT_l^*(\mathcal{D}) = Nt_0$. Thus,

$$T_l - T_f \leq 1 + (3n+1)t_0$$

Theorem 2 *Let $Alg = (J, D)$, where $J \equiv (A, Nb)$, be a parametrized UDA for which a schedule exists. Let n be the problem dimension. Then*

$$T_l - T_f \leq 1 + (3n+1)t_0$$

for $N \geq 3n+1$, where t_0 is the time of the optimal linear schedule for the unit domain $J \equiv (A, b)$. The optimal linear schedule and the free schedule differs by a constant independent of the size of the domain. The constant that we give only depends upon the shape of the domain (the constraint matrix A and the vector b) and upon the dependence vectors (the dependence matrix D).

Example: Consider a n-dimensional parallelepiped of size N and let α the largest component in absolute value of the dependence vectors. For the cube of size 2α, which is fat, $t_0 = T_l \leq 2n\alpha$. Thus for $N \geq 3n+1$, $T_l - T_f \leq 1 + 2n(3n+1)\alpha$ at worst. In practice, the difference is often smaller. A direct proof for the particular case of a cube could provide more precise results.

4 Conclusion

We view the contribution of this paper as a link between two important works:

- Karp, Miller and Winograd [5] have studied computable URAs over a particular domain "F_n". They have established a local relation in each computation point between the free schedule and a piecewise linear schedule.
- Shang and Fortes [17] have developed a procedure that permits to determine the best linear schedule vector over general convex domains $J \equiv (A, b)$.

We have shown that for an arbitrary convex domain $J \equiv (A, Nb)$, sufficiently fat, the difference $T_l - T_f$ between the total execution time of the best linear schedule and that of the free schedule is bounded by a constant independent of the parameter size N. Since linear schedules are very simple and easy to use in practice, it is very stimulating to know that they are close to optimality.

Acknowledgments

We would like to thank Apostolos Gerasoulis for hosting Alain Darte during his stay at Rutgers and for fruitful discussions.

References

[1] U. Banerjee, "An introduction to a formal theory of dependence analysis", *The Journal of Supercomputing 2*, 1988, pp. 133 - 149.

[2] V. van Dongen and P. Quinton, P, "Uniformization of linear recurrence equations: a step towards the automatic synthesis of systolic array", *Proceedings of the International Conference on Systolic Arrays*, K. Bromley et al. eds., IEEE Computer Society Press, 1988, pp 473 - 482.

[3] M.L. Dowling, "Optimal code parallelization using unimodular transformations", *Parallel Computing*, 16, 1990, pp. 157 - 171.

[4] J.A.B. Fortes and F. Parisi-Presicce, "Optimal linear schedules for the parallel execution of algorithms", *Proceedings of the 1984 International Conference on Parallel Processing*, August 1984, pp. 322-328

[5] R.M. Karp, R.E. Miller and S. Winograd, "The organization of computations for uniform recurrence equations", *J. ACM*, Vol. 14, No. 3, July 1967, pp. 563 - 590.

[6] S.Y. Kung, *VLSI array processors*, Prentice-Hall, 1988.

[7] L. Lamport, "The parallel execution of DO loops", *Commun. ACM*, Vol. 17, No. 2, February 1974, pp. 83 - 93

[8] D.I. Moldovan and J.A.B. Fortes, "Partitioning and mapping algorithms into fixed-size systolic arrays", *IEEE Transactions on Computers*, Vol. 35, No. 1, January 1986, pp. 1 - 12.

[9] J.K. Peir and R. Cytron, "Minimum distance: a method for partitioning recurrences for multiprocessors", *IEEE Transactions on Computers*, Vol. 38, No. 8, August 1989, pp. 1203 - 1211.

[10] C.D. Polychronopoulos, "Compiler optimization for enhancing parallelism and their impact on architecture design", *IEEE Transactions on Computers*, Vol. 37, No. 8, August 1988, pp. 991 - 1004.

[11] C.D. Polychronopoulos, *Parallel Programming and Compilers*, Kluwer Academic Publishers, Boston, 1988.

[12] P. Quinton, "Automatic synthesis of systolic arrays from uniform recurrent equations", *Proceedings 11th Annual Symp. Computer Architecture*, IEEE Computer Society Press, 1984, pp. 208 - 214

[13] P. Quinton and Y. Robert, *Algorithmes et Architectures Systoliques*, Masson, Paris, 1989.

[14] T.R. Rockafellar, *Convex Analysis*, Princeton University Press, NJ, 1970

[15] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, New York, 1986

[16] W. Shang and J.A.B. Fortes, "Time optimal linear schedules for algorithms with uniform dependences", *Proceedings of International Conference on Systolic Arrays*, K. Bromley et al. eds., IEEE Computer Society Press, 1988, pp. 393 - 402.

[17] W. Shang and J.A.B. Fortes, "Time optimal linear schedules for algorithms with uniform dependencies", *IEEE Transactions on Computers*, Vol. 40, No. 6, June 1991, pp. 723-742.

[18] W. Shang, M.T. O'Keefe, J.A.B. Fortes, "On loop transformations for generalized cycle shrinking", Proceedings of International Conference on Parallel Processing, August 1991, pp. II-132 - II-141.

[19] M. Wolfe, *Optimizing Supercompilers for Supercomputers*, MIT Press, Cambridge MA, 1989.

[20] M. Wolfe, "Data dependence and program restructuring", *The Journal of Supercomputing 4*, 1990, pp. 321 - 344.