Some Efficient Solutions to the Affine Scheduling Problem. Part II. Multidimensional Time

Paul Feautrier

Received October 1992; Revised June 1993

This paper extends the algorithms which were developed in Part I to cases in which there is no affine schedule, i.e. to problems whose parallel complexity is polynomial but not linear. The natural generalization is to multidimensional schedules with lexicographic ordering as temporal succession. Multidimensional affine schedules, are, in a sense, equivalent to polynomial schedules, and are much easier to handle automatically. Furthermore, there is a strong connection between multidimensional schedules and loop nests, which allows one to prove that a static control program always has a multidimensional schedule. Roughly, a larger dimension indicates less parallelism. In the algorithm which is presented here, this dimension is computed dynamically, and is just sufficient for scheduling the source program. The algorithm lends itself to a “divide and conquer” strategy. The paper gives some experimental evidence for the applicability, performances and limitations of the algorithm.

KEY WORDS: Loop nest scheduling; affine computation; automatic parallelism detection; parallelizing compilers.

1. INTRODUCTION

In the first part of this paper, I have presented a new algorithm for computing affine and piecewise affine schedules for Generalized Dependence Graphs and affine systems of recurrence equations. The algorithm is simple and efficient. However, there are programs and systems which do not have such a schedule. This is equivalent to the observation that there are

1 Laboratoire MASI, Université de Versailles St-Quentin, 45 Avenue des Etats-Unis, 78035 Versailles Cedex France. (e-mail: feautrier@masi.uvsq.fr).
programs which cannot be executed in linear time on a paracomputer, and should not come as a surprise.

An example of such a program is given in Fig. 1. Application of the methods of Part I shows that this program—hereafter referred to as program 1—has no affine schedule. It is possible to prove that its free schedule is:

\[
\theta_1(i, j) = \frac{i(i+1)}{2} + j
\]  

(1)

which has a mean degree of parallelism of 1. Hence, the original program is totally sequential.²

However, the automatic construction of polynomials schedules seems to be beyond present day techniques. Examination of program 1 and other similar examples suggests another solution: the use of multidimensional time. This is nothing out of the way: a clock with two hands define a twodimensional time, each hand being associated with one dimension. The order on such a time is lexicographic ordering. Program 1, for instance, has the following two-dimensional schedule:

\[
\theta_2(i, j) = \binom{i}{j}
\]  

(2)

which will be seen later to be equivalent to schedule (1).

For usual clocks, the minor components of time are always uniformly bounded. As a consequence, their time may be linearized if necessary. This restriction is not enforced for multidimensional schedules, as the preceding example shows.

² For definiteness, I will suppose that + in this program stands for some operator with no special algebraic properties. As a consequence, the computation must be executed as written; there is no possibility of sharing the work between processors by taking advantage, e.g. of associativity. This remark will stand for all examples in this paper. Computing schedules for operators with nontrivial algebraic properties is a largely open problem.