Solution to the Affine Scheduling Problem

Affine scheduling is way in which the set of operations and the dependence graph are described as the set of linear inequalities. This works if the number of operations is small but in practice the size of **Ω** is equal to the total operation count of the program. So the paper present how to take in to account the special form of usual dependence graphs in order to compress resulting inequalities into set of inequalities whose size does no longer depend on the program complexity and then solved by efficient algorithm.

First is to decide whether a schedule exists that means that if a GDG is consistent i.e. there is no vertex with infinite incoming path.

The first method is called Vertex method

1. Compute the generating system for all polyhedral D_s , S € V and Re , e in the set of edges.

2. Write the equations in the form <x,y>€ Re $\Rightarrow$ **$\theta(\delta(e),y)$>= $\theta(\sigma(e),x)+1$** at all vertices of Re and also write instances of **$\theta(S,x)$>=0** at all vertices od Ds.

3. Solve the resulting finite system of linear in equalities by any standard algorithm.

But also selecting a good schedule is problem as it not possible to enumerate all casual schedules, one of the process is by inspection which is quite wasteful , other in which one picks the performance factor and tries to optimize over the solution space. This can be done by PIP software as per details provided in the paper.

Moreover if the schedules lack details such as orderings of the domain or Farkars multipliers , more definite results are needed so we come to the Minimum Latency Schedule. This is the special case where all iteration domains are bounded, so one can define the maximum latency of the all schedules and then tries to find the minimum latency schedule. But this method has defect if the there are several structure parameters, and then solution will depend upon their orderings.

Other is Boundary delay schedules, i.e. for all edges of the DGD lies between 1 and a positive integer **δ.** But here minimum schedule is not unique and also there exist GDG for which bounded delay does not exist.

So to sort out these problem , paper presents the Dual method, i.e. by finding the best affine schedule and the ordering on schedules is point wise ordering.

$\theta_1 < \theta_2 \equiv \forall$ u € Ω : $\theta_1(u)$<= $\theta_2(u)$

and if **θ_1, θ_2** satisfy the causality condition then **$\theta_3(u)=\min(\theta_1(u) ,\theta_2(u))$.** Since affine functions are concave and the min operator converse concativity, **θ** is concave; it will be called best concave schedule.

The method is quite fast, and no enumeration is needed, and also is not limited to the uniform dependencies but also solves the problems with unbounded domains.

Questions

1. How structure parameter of graph is computed?

2. T=Card **Ω**. What is Card?

3. Convex , concave schedules and sub-optimal schedules?

Some Efficient Solutions to the Affine Scheduling Problem. I. One-Dimensional Time

The basic problem of parallel programming can be expressed in the following terms. We are given a set of operations with a dependence relation on them to indicate the order of some operations. To construct a parallel program we have to select a partial order which satisfies the relation. In general we are not interested in one particular dependence graph but in a possible infinite family of similar graphs - each specified by one or more integer parameters. A parallel program is described as a scheduling function from the set of operations to the set of numbers. The difficulty is that this problem is NP-complete. Therefore we restrict the possible solutions to affine functions. We want to create a scheduling function with minimal latency.

The first point is to decide whether a schedule exists which means there is no vertex u with an infinite incoming path. This would be that u is executed after infinite operations. The next point is that a system of uniform recurrence equations which is consistent is decidable while the problem is not decidable for a nonuniform graph from either an infinite family or with one infinite domain. With this in mind we can only try to approximate the best schedule with techniques described in the following.

To compute a schedule we start first with a prototype schedule which describes how the scheduling function should look like. Then we extract inequalities from the dependence graph and try to solve them. We can solve this by any standard algorithm. The problem is that the number of vertices we have to use for a hypercube in p-space can be 2^p while the describing inequalities may only be 2*p. To solve this problem we use an other method. We use Farkas Lemma to extract an affine form from a nonempty polyhedron. With all these equations we use a kind of Gauss-Jordan elimination to reduce the number of unknown variables.

The next problem is to select a good schedule from these (in)equations. We can try to extract a minimum latency schedule - as discussed before or we can go for a bounded delay which bounds the maximum delay from two statements in a dependence by a constant. This is useful to get good cache hits within the application. An other approach is the dual method where we extract two schedules and calculate our new schedule as the minimum of both (point wise). This is also an acceptable schedule.

The advantage of this method is that it doesn't need an enumeration and it's not limited to uniform dependencies. Ultimately, the solution is found by solving a parametric integer program of relatively small size which can already be done efficiently.

Open questions:
1) What does the Farkas Lemma say?
2) Why is the Farkas Lemma useful?
3) How to generate a generating system for a polyhedra?
4) Is this used in practice?

Summary of “Some efficient solutions to the affine scheduling problem One-dimensional Time”

January 7, 2013

1 Summary

The paper mainly solve the problem of finding closed form schedules as affine or piecewise affine functions of the iteration vector. The newly developed method constructs affine schedules without enumeration fast. An efficient algorithm is presented reducing the scheduling problem to a parametric linear program of small size, which can be readily solved by an efficient algorithm.

In this paper, we define the Generalized Dependence Graph (GDG) of programs, and schedules of a GDG. In order to compute affine schedules, we develop vertex method and we have the farkas algorithm. It is not possible to enumerate all causal schedules, so we have to device some criterion and select the best one. We can have minimum latency schedules which requests us to define a total latency: the maximum value of all schedules, and we try to find a minimum latency schedule. Meanwhile, we can use bounded delay schedule, which is more constrained than minimum latency schedules. However, the minimum delay schedule is no unique. Uniform recurrences has the constant delay, hence we don’t need Farkas Lemma for the expression of the causality condition.

The main advantage is that no limit to uniform dependences. But a DFG has many uniform dependences. We use some devices to eliminate some Farkas multipliers which make unknowns only in positive or negative occurrences. The solution is computed by solving a small size parametric integer program. We use PIP software to do this. In example 3.4, the upper limits of the domain can be removed without changing the result. We try for a bounded delay schedule or a best concave schedule. For concave schedule, the result has minimum latency. When DFG is uniform, the latency of the best concave schedule is asymptotically optimal. The method has two weakness. Due to some programs with non-concave free schedule, we cannot find all piecewise affine schedules. Importantly, there are some GDG without affine schedule which is given as the example in Fig.9. In this case, we can verify the example is unfeasible. We know an affine schedule has a latency which is linear in the structure parameters.

Hence, this example program has no parallelism, and has the quadratic running time $n^2/2$.

2 Questions