==========
Summaries for the sixth meeting of the seminar "Automatic Parallelization in the Polyhedral Model"
Topic: "Some Efficient Solutions to the Scheduling Problem. Part II. Multidimensional Time"

==========

Summary 1 of 3

We discussed last time a new algorithm for computing affine schedules with only one dimension. The algorithm was simple and efficient. However, there are programs and systems which do not have such a schedule. This is related to the observation that there are programs which cannot be executed in linear time on a paracomputer because of dependencies. Therefore we need to use multidimensional time to describe the execution of such programs. With such schedules it is possible to prove that the DFG of a sequential program always has a multidimensional affine schedule. We can also transform a multidimensional schedule to a one dimensional schedule where every statement is executed at time t where t is the number of statements executed before. But this schedule does not have to be affine. In fact any instance of a static control program has a linear schedule.

To construct a multidimensional schedule we try to define the scheduling function piecewise. We start with the first component and use the Farkas algorithm in combination with the linear constraints. If we've solved that problem we have to check for every edge if the delay in the first component is greater than zero. If this is the case we are finished and have found and one dimensional schedule. Otherwise we need to select in some way a particular solution and do the same algorithm for the next component until all edges have a delay of at least 1. This algorithm is complete and correct but the price to pay is the highly nonlinear character of the algorithm. Therefore the solution method is basically an enumeration.

There is a greedy algorithm which wants to minimize the dimension of the schedule because a schedule of dimension d will have most of the time(!) a latency of O(n^d). With this in mind a smaller schedule would be always better but this is not always the case. The goal of this algorithm is to satisfy as many edges as possible at each stage of the solution. To do this we have (at every stage) a set U of unsatisfied edges and our aim is the determination of the next component of the schedule in such a way that as many edges of U as possible are satisfied. This algorithm is correct but no longer complete. There is also a dual version of this greedy algorithm. In the first part we used a two-step process: determine first the set of satisfied edges then find the best concave schedule. But it is also possible to solve this problem with just one linear program.

Open questions:
1) definitions / formulas (p.398)
2) e $in U and e $nin U are complements? (p.397)
3) used in practice?

==========

Summary 2 of 3

This paper answers the question what we will we do when there is program for which there is no affine schedule i.e. problems whose complexity is polynomial. It presents a algorithm based on´divide and conquer ‘method which computes the dimension automatically and sufficient for scheduling the source program.

Construction of this type of schedule considers multidimensional time, just like two hands of the clock, each acts like one dimensional time and order on such a time is lexicographically ordering.
First we define the causality condition or another possibility

Secondly, we find the delay associated with the edge , whose components are affine.

And lastly
If solution is found and the original GDG has one dimensional schedule, if not we propose to select in some way particular solution of the problem $G\begin{pmatrix}\mu^{(1)}\\ \lambda^{(1)}\end{pmatrix} \geqslant d$.

There could be another possibility that we partition the set of edges in GDG into and $e \notin U^{(1)} \Rightarrow \Delta_e(y)[1] > 0$.

And the next component of the schedule must satisfy $y \in \mathcal{P}_e^{(1)} \Rightarrow \Delta_e(y)[2] \geqslant 0$ and is void when since $\mathcal{P}_e^{(1)}$ is polyhedral, so algorithm proceeds until all

are empty that means d-dimensional schedule is found other no solution exits. And since this algorithm has very high complexity.

Now paper introduce simpler version of the algorithm, which is no longer complete. It applies greedy algorithm whose aim to reduce the dimension of the schedule. Here we let U be the set of unsatisfied edges and we try to find the next component of the schedule such that as many edges in U is satisfied.

Next if we assume that we have found K components of a multidimensional schedule and that the set of unsatisfied edges is $U^{(k)}$.The greedy algorithm will terminate the

proof if, the solution of the linear program satisfy at least one edge of $U^{(k)}$. Since solution of linear program is unique and feasible if it satisfy at least one edge of U.

Question
1.)Correct generalization of causality condition? How?

==========

Summary 3 of 3

The journal article by Paul Feautrier continues the topic from its predecessor, now extending the solution to the multi-dimensional case. This transition is motivated by the fact that single- dimensional schedules were not able to give a schedule for every type of dependency graph. In particular, the solution had to be affine. Multidimensional schedules are not as constrained and are able to handle a polynomial polyhedral complexity. Most notably, any static control program has a multi-dimensional schedule. This article largely deals with proving the previous statement and the development of an (efficient) algorithm to compute such a solution.

In section 2, the author introduces an appropriate variant of the causality condition. In order to extend the condition from the one-dimensional case, the authors use the lexicographic ordering. This definition also gives a natural extension to the definition of latency, which is then simply the cardinality of the schedule. The author then proceeds to prove that there exists a multi-dimensional schedule for any static control program. Furthermore, any static control program has a linear schedule, which however might provide no parallelism. We are left with the problem of finding an optimal schedule.

Hence, the next goal is the construction of an algorithm to find an optimal schedule. The process is very similar to the one-dimensional case. The delay is similar to the one-dimensional case, and each component is an affine form of the iteration vector and parameters. The components of the schedule are also restricted to be positive (as well as the first component of the delay). Once again applying the farkas algorithm once gets a set of solutions, in which we can either select a feasible solution. In this case, the algorithm found a one-dimensional schedule. If not, the search is continued by partitioning the set of edges. The algorithm will then recursively enumerate the set of solutions by fixing the first component depending on the split, and checking for solutions for the next component.

The problem of the hereby described algorithm is that it basically enumerates the whole search space until a solution is found. The next step is a greedy algorithm, which will try to minimize the dimension, as there is a causal relation between dimensionality and complexity of the solution (though the author also points out that this is not always true). In order to reduce the dimension of the final solution, the algorithm will always try to make the split such that as many edges as possible are already satisfied. This is achieved by a new LP, which gives a set of satisfied edges. Using a primal- dual scheme, this solution can be improved to find the best concave schedule.