A Practical Parallel Facility for Dynamic Programming

Danilo Bruschi
Giovanni Pighizzini
Dipartimento di Scienze dell'Informazione
Università degli Studi
Via Comelico 39, 20135 Milano – ITALY
{bruschi,pighizzi}@imiucca.csi.unimi.it

Submitted to 3rd DIMACS challenge

1 Introduction

Dynamic programming is a technique applied for solving many combinatorial problems. The solution of a problem instance is determined by evaluating all subinstances, of the given instance, in a bottom–up fashion; i.e., initially the smallest subinstances are evaluated, in subsequent steps their solutions are combined in order to obtain answers to instances of increasing size until the solution of the original instance is determined.

In many occasions such a strategy permits of saving time with respect to top–down techniques, which instead may work on identical subinstances a lot of time. In particular, dynamic programming is often used to solve optimization problems that satisfy the *principle of optimality*, that can be expressed in the following form: the optimal solution can be written as a recurrence equation in solutions to subproblems.

Even if many algorithms textbook describe the dynamic programming technique independently by evaluation and exploring graph techniques such as depth–first search, backtracking, branch–and–bound and so on, it is not

difficult verify that dynamic programming can be thought of as one of such graph (tree) search technique. In fact it is well known that a wide class of dynamic programming problems can be represented by a system of equations of the following form:

$$\begin{aligned} c(i,i) &= init(i), 0 \leq i \leq n; \\ c(i,j) &= \oplus_{i<m\leq j}(c(i,m-1) \otimes c(m,j)), 0 \leq i < j \leq n, \end{aligned}$$

where *init* is a given function which computes costs of "primitive" subproblems and $\oplus, \otimes$ are operations defined over a semiring, as for instance the natural semiring $S = (N, +, max, 0, 1)$ in the case of optimization problems.

Now it turns out that such a description defines what is usually defined in literature *implicit directed acyclic graph* or *circuit*. This graph has the following structure. The nodes of this graph are leaves or are marked as $\oplus$ or $\otimes$ nodes. Each node is labeled by a pair $(i,j), 0 \leq i < j \leq n$. Leaves are given by all elements labeled by $(i,i), 0 \leq i \leq n$. Nodes $\otimes$ labeled by (i,j) are connected to nodes $\oplus$ labeled $(i,m-1)$, (m,j), nodes $\oplus$ labeled by (i,j) are connected to nodes $\otimes$ labeled (i,j). Given such a graph, the dynamic programming technique is then reduced to evaluating as fast as possible the *solution tree*, i.e., the set of gates which contribute to the computation of the final solution.

2 Our Proposal

Up to now, a certain number of efforts have been realized from the theoretical point of view for obtaining parallel versions of some graph search procedure but to our knowledge very few efforts have been spent both from the theoretical and practical point of view for defining specific dynamic programming parallel techniques. On the other hand, in sequential computations the range of applicability of dynamic programming is at least as wide as those of the techniques just mentioned. Thus from our point of view the problem of parallelizing dynamic programming deserves more investigations. *Such investigations with a particular emphasis on implementability aspects of the solution proposed, are the subject of our research.*

In order to get such a result we think of following this strategy.

In previous section we have shown that a dynamic programming problem can be reduced to the problem of evaluating a circuit over a suitable semiring. It turns out that even if no specific general algorithms have been proposed for parallel dynamic programming, a certain number of researchers have however afforded the problem of efficient circuit evaluation. In particular in [MR85, MRK88] efficient parallel algorithms for evaluating circuits over commutative semiring using CRCW PRAMs are proposed. These algorithms have been applied for obtaining theoretical efficient parallel solutions to some well known problems such as context–free languages recognition [Ryt86, BP93]. Quite recently these algorithms have also been extended to work on a non–commutative semiring in order to solve some problems related to data retrieval [ABP91].

Thus we think of pursuing our objective referring to the results just mentioned. More precisely this is our high–level scheduling:

- to adapt and implement the algorithms described in [MRK88, ABP91] on real architectures;
- to use the results of the previous step for implementing generic routines for parallel dynamic programming;
- to use the routines developed in previous step for writing solutions to real problems such as Minimum Distance Error–Correcting Parsing, Optimal Search Trees construction, Shortest Paths.
- to measure the performance of such solutions and compare them with those obtainable by sequential implementations or other parallel implementations [Vel86]. With respect to input generators we are looking for suitable random graph generators for the last two problems. With respect to the first problem we are still discussing about the target language, in particular we are deciding about C or Fortran. We think of individuating benchmark instances after a while the routines we are going to develop will work.

The entire experiment will be realized using a CM 200 with 8K processors, We are also investigating the possibility of accessing different architectures in order to facilitate a better understanding of the impact of the hardware architecture on the algorithms implemented.

References

[ABP91] E, Allender, D. Bruschi, and G. Pighizzini, "The complexity of computing maximal word functions", Technical Report 92–15, DIMACS, 1992. To appear in '*Computational Complexity. Proc. 8th FCT Conference*: LNCS **529**, 157–168, 1991.

[BP93] D. Bruschi, and G. Pighizzini, "A general context–free parser", *Manuscript*, 1993.

[MR85] G. Miller and J. Reif, "Parallel tree contraction and its application," *Proc. 26th FOCS*: 478-489, 1985.

[MRK88] G. Miller, V. Ramachandran and E. Kaltofen, "Efficient parallel evaluation of straight–line code and arithmetic circuits," *SIAM J* **17**: 687–695, 1988.

[Ryt86] W. Rytter, "On the complexity of parallel parsing of general context–free languages," *TCS* **47**: 315–321, 1986.

[Vel86] M. Veldhorst, " Parallel dynamic programming algorithms", *Proc. 2nd ConPar Conference*: LNCS **237**, 393–402, 1986.