

# A SOFTWARE PIPELINING METHOD BASED ON A HIERARCHICAL SOCIAL ALGORITHM

Felipe Fernández\* , Abraham Duarte\*\* and Angel Sánchez\*\*

Felipe.fernandez@es.bosch.com, a.duarte@escet.urjc.es, an.sanchez@escet.urjc.es,

(\*) Dept. Fotónica, FI-UPM, Campus Montegancedo, 28660 Madrid, Spain

(\*\*) ESCET-URJC, Campus de Móstoles, 28933 Madrid, Spain

**Abstract:** Software pipelining is a compile-time scheduling technique that overlaps successive loop iterations to achieve instruction-level parallelism. It allows us to hide memory latency by overlapping the prefetches for a future iteration with the computation of the current iteration. This paper presents an efficient algorithm for determining the iteration bound of cyclic data flow graphs and the optimal loop scheduling. To solve this problem, a new general metaheuristic has been developed. This metaheuristic is inspired in some hierarchical social structures and processes. Several variant of the proposed algorithm has been implemented and tested on some benchmark programs.

**Key words:** Metaheuristics, software pipelining, extended retiming, scaled retiming

## 1. INTRODUCTION

In parallel processing, Data Flow Graphs (DFG) are frequently used to specify real time applications performing repetitive tasks, such as signal and image processing, and to express the available concurrence. DFG's are directed graphs, where directed edges model the precedence constraints between nodes. Each node accepts one input data from each of its input node, executes some tasks and outputs one data to each of its output nodes. External data to the system are sampled periodically; the corresponding period is called the iteration period. This parameter is also the time period of a DFG between two iterations of a node. System throughput is improved by reducing the sampling period as more processors are used to increase concurrency. The shortest iteration period is called iteration bound and the corresponding scheduling is termed rate-optimal. A DFG contains a number of circuits that determine the iteration bound, these are named critical

circuits. An active research goal has been to achieve the maximum throughput with the minimum number of processors using the appropriate scheduling. There are two main aspects of scheduling: timing and processor allocation. Each node must be assigned a starting time and a processor to execute it; this scheduling optimization is in general an NP-complete problem. References [2-9] have discussed various scheduling methods to achieve the minimum iteration period, called maximum-throughput scheduling with unlimited resources. This work is concerned only with the unlimited resource case.

Scaled retiming techniques on a Data Flow Graph (DFG) [34] are here used to model the static schedule of a DFG, reorganize the scheduling of an iteration and solve the corresponding software pipelining or loop scheduling problem.

Hierarchical social (HS) algorithms [24] are also introduced in this paper as a new metaheuristic [25-30] for finding optimal arrangements in a discrete solution space. HS algorithms are inspired in the hierarchical social structures and processes in order to solve multilevel discrete optimization problems. HS algorithms make use of local search heuristics into a group-population-based strategy. This metaheuristic is utilized to efficiently solve the considered loop-scheduling problem.

The rest of the paper is organized as follows: Section 2 outlines the fundamentals of HS algorithms. Section 3 briefly explains the basic model used. The scheduling methods of cycle and cocycle are described in Section 4. Section 5 presents a bilevel linear program to solve the corresponding scheduling problem. Section 6 shows an efficient HS scheduling algorithm. Some variants of the previous algorithm are discussed in Section 7. Section 8 shows an application example. Some benchmark results are given in Section 9. Finally we conclude this paper in Section 10.

## **2. SCHEDULING ALGORITHM**

The considered Data Flow Graph (DFG) scheduling problem, with unlimited resources, is modelled in this paper by the slack graph [3]:

$$G_{\alpha} = \{V, A, \alpha[D]-[T]\} = \{V, A, [W_{\alpha}]\}$$

where  $V$  is the ordered set of nodes that represents the set of tasks ( $|V|=n$ ),  $A$  is the set of oriented edges or arcs of the corresponding graph that represents the set of precedence relations among the nodes ( $|A|=m$ ),  $\alpha$  is the pipelining period,  $[D]$  is a column delay  $m$ -vector (algorithmic delays) and  $[T]$  is a column computation-time  $m$ -vector (architectural delays).

The scheduling algorithm here considered generalizes max-plus Howard's algorithm [1], which has been ranked first in different empirical studies [2]. We also present an intuitive social explanation of the corresponding algorithmic process. This way, the strategy used in Howard's

## Software pipelining method based on a hierarchical social algorithm

algorithm is generalized and interpreted as a hierarchical social (HS) evolutionary metaheuristic [4], and the associated local search technique is easily extended to other optimization problems.

This scheduling algorithm can be defined by the bilevel integer linear program:

$$\begin{array}{l} \text{Min } \alpha \\ \text{st: } \left\{ \begin{array}{l} \text{Min } ([1,1,\dots,1][R]) \\ [R] \\ \text{st: } [DF]=[C][R] + \alpha [D]; [DF] \geq [T]; [R] \geq 0; \alpha \in \mathbb{Z}; [R] \in \mathbb{Z}^n \end{array} \right. \end{array}$$

where  $\alpha$  is the pipelining period,  $[R]$  is a column potential retiming  $n$ -vector,  $[1,1,\dots,1]$  is a row  $n$ -vector,  $[C]$  is the corresponding arc-node incidence  $m \times n$ -matrix and  $[DF]$  is the final column delay  $m$ -vector obtained

In the considered HS scheduling algorithm, the set of feasible individuals are the vertices  $V$  of the graph, the set of feasible relations are the arcs  $A$  of the graph, and the set of feasible social solutions are called policy graphs  $\Pi$ . The individuals objective function is the retiming vector  $[R]$  and the final group objective function is the pipelining period  $\alpha$ . A feasible social state is characterized by its *policy graph*  $\Pi$  that defines the set of active relations in each stage. This policy graph contains all the nodes of the original DFG and verifies that the outdegree of each node is equal to one. In general it is composed by a set of connected components or group graphs, and each group is composed by linked individuals. Each group graph  $G_k$  is divided into two parts: core circuit and periphery. The core circuit determines the value of the corresponding pipelining period  $\alpha_k$  of the involved group. The periphery is the acyclic part, and consists of a set of disjoint directed trees with the out-root placed in a node of the corresponding core circuit.

The group and individual objective functions are hierarchically optimized through the local modification of the corresponding policy graph.

In the HS metaheuristic, the neighbourhood of a vertex  $i$ , to carry out the corresponding local search, is defined by the corresponding forward star  $FS(i)$  (out-arcs) and by the out-neighbourhood  $N^+(i)$  (out-vertices):

$$FS(i) = \{(i,j) \in A\}; \quad N^+(i) = \{j \in V : (i,j) \in A\}$$

For each group  $G_k$ , there are two main local search strategies: winner strategy and loser strategies that depend on the relative value of the corresponding group objective function or group pipelining period  $\alpha_k$ . The groups that have the highest value of  $\alpha_k$  are classified as winner groups and the rest are classified as loser groups. The output relation  $\Pi(i)$  of each individual or vertex  $i$  is modified according to loser or winner condition of the group that belongs:

**If** (vertex  $i \in \text{Loser\_Groups}$ ) **then**  $\Pi(i) = \arg \max_j \{\alpha_j : j \in N^+(i)\}$

**If** (vertex  $i \in \text{Winner\_Groups}$ ) **then**  $\Pi(i) = \arg \max_j \{r(j) - (\alpha_j d(a) - t(a)) : a \in FS(i)\}$

The search strategy of loser groups (smaller pipelining period) tries to activate their relations with a group with higher pipelining period and the local search strategy of nodes of a winner group tries to activate the relation that improves their individual objective function.

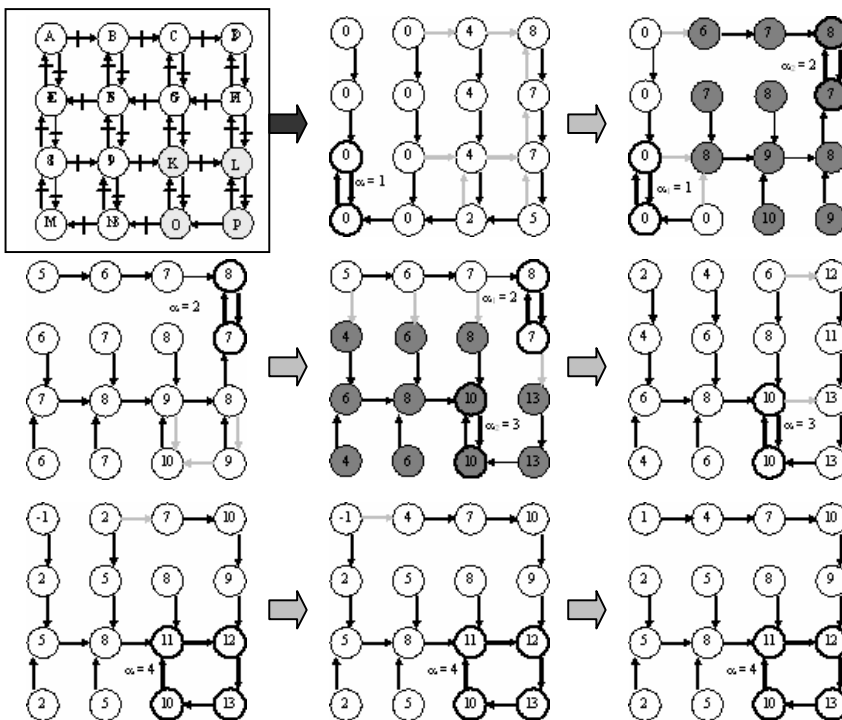
This way, the loser groups tend to disappear in the corresponding evolution process. After the relations modification of a policy graph, it is necessary to compute the new values of  $\alpha_k$  and actualize the corresponding potentials  $R$  in each group  $G_k$  of a policy, visiting all the nodes of each group in backward topological order, and setting an arbitrary node of each group (connected component) as reference potential. For each  $a=(i,j)$  of the policy graph, we have

$$\alpha(i) = \alpha(j); \quad r(i) = r(j) - (\alpha_j d(a) - t(a))$$

### 3. OPTIMAL SCHEDULING EXAMPLE

The initial graph of Figure 1 shows the strongly connected DFG considered. In this graph: circles represent tasks; arcs represent precedence relations and black solid rectangles represent algorithmic delays. All DFG arcs have an architectural delay equal to 1 unit, except the output arcs of grey nodes (K, L, O, P) that have a computational delay of 3 units.

The rest of graphs of Figure 1 shows the consecutive policies applied: all individual objective functions  $R$  are displayed inside the nodes; the group objective functions  $\alpha_k$  are shown near to the circuit of each group; if there are several groups, the nodes of each group are represented with different grey levels. For each algorithm stage, the next active relations are shown in grey colour.



## Software pipelining method based on a hierarchical social algorithm

Figure 1. Original DFG example and policy graphs derived in the HS evolution

The optimal pipelining period  $\alpha = 4$  and retiming  $R^T = [1, 4, 7, 10, 2, 5, 8, 9, 5, 8, 11, 12, 2, 5, 10, 13]$  are derived from the last policy graph of Figure 1

To get a normalized scheduling, all the potentials are referred to the largest potential. This way, the first iteration scheduling for each node  $u$  of the DFG is given by the following expression

$$S(u) = -R(u) + \max\{R\} = -R(u) + 13 \quad u \in V$$

## 4. PERFORMANCE ANALYSIS AND CONCLUSIONS

The algorithm behaviour is summarized in Table 1, where  $cc$  denotes the number of arcs of the critical circuit. The benchmarks (sXXXX) are from ISCAS 89/93 [5] and (AbrXXXX) are specifically developed for checking the algorithm capability to solve a more complex circuital structures.

The last tree columns of Table 1 show the number of iterations of the best-improvement local search implementation of algorithm variants  $H1$ ,  $H2$  and  $H3$

- $H1$ : it uses a *non-autonomous strategy*, where each group evolves independently only one iteration.
- $H2$ : it uses a *fixed autonomous strategy*, where each group evolves independently a number of iterations *equal* to (1/2 /3).
- $H3$ : it uses a *variable autonomous strategy*, where each group evolves independently a number of iterations *less or equal* to (1/2 /3).

These experimental results show a number of iterations approximately linear with the number of nodes or arcs. Moreover, we can conclude that in general, more elaborated HS algorithms, as  $H3$  are better adapted for complex solution spaces.

Table 1. Performance comparison of variants (H0, H1, H2) of the HS scheduling algorithm.

Benchmark	$n$	$m$	$cc$	$\alpha$	H1	H2 (1/2/3)	H3 (1/2/3)
s3330	1612	2739	4	7	11	10/9/12	10/10/10
s3271	1902	3050	10	28	8	10/9/12	10/10/10
s3384	1754	2850	3	1	8	8/9/12	8/8/8
s6669	2407	4156	60	64	10	10/12/12	10/10/10
s4863	3218	5547	8	18	9	8/9/8	8/8/8
Abr0010	100	270	4	4.33	33	18/24/32	18/18/18
Abr0020	1200	9159	2	10	29	18/18/24	18/18/18

## REFERENCES

1. J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. MC Gettrick, J. -P. Quadrat, "Numerical computation of Spectral Elements in Max-Plus Algebra", *IFAC Conference on Systems Structures and Control*, Nantes, France, Jul. 1998

2. A. Dasdam, S. S. Irani and R. K. Gupta, "Efficient Algorithms for Optimum Cycle Mean and Optimum Cost to Time Ratio Problems", *Proc. 36<sup>th</sup> Design Automation Conference*, 1999.
3. F. Fernández, A. Sánchez, A. Duarte, An Optimal Software-Pipelining Method for Instruction-Level Parallel Processors based on Scaled-Retiming, In Proc. ISPA-01, Vol. I, pp.405-410, Pula, Croatia, 2001.
4. F. Fernández, A. Duarte, A. Sánchez, Hierarchical Social Algorithms: A New Metaheuristic for Solving Discrete Bilevel Optimization Problems, Submitted to European Journal of Operational Research, Feature Issue on Metaheuristics in Multiple Objective Optimization, December 2003.
5. ISCAS'89/93 Benchmark Informat. [http://www.cbl.ncsu.edu/CBL\\_Docs/iscas89.html](http://www.cbl.ncsu.edu/CBL_Docs/iscas89.html), 1989. Sequential Benchmark Circ., <http://www.visc.vt.edu/~mhsiao/iscas93.html>, 1993