ELSEVIER

Computers and Chemical Engineering 25 (2001) 997–1002

Computers & Chemical Engineering

www.elsevier.com/locate/compchemeng

Parallel observability analysis on networks of workstations

Ignacio Ponzoni [a,b], Gustavo E. Vazquez [a], Mabel C. Sánchez [b], Nélida B. Brignole [b,*]

[a] *Department of Computer Science, Universidad Nacional del Sur, Av. Alem 1253, 8000 Bahía Blanca, Argentina*
[b] *Planta Piloto de Ingeniería Química (UNS-CONICET), Complejo CRIBABB, Km. 7, Camino La Carrindanga, CC 717, 8000 Bahía Blanca, Argentina*

Received 15 February 2000; received in revised form 27 December 2000; accepted 28 December 2000

Abstract

In this work we present the parallelisation of the global strategy with first least-connected node (GS-FLCN), which is a novel structural technique for the classification of unmeasured variables in process plant instrumentation design. The algorithm aims at partitioning the process' occurrence matrix to a specific block lower-triangular form. A parallel master–workers philosophy is employed to search for all the paths of a given length existing in the associated graph. The code was conceived for distributed environments and the implementation was carried out using the parallel virtual machine (PVM) library. The performance of the parallel algorithm was tested for industrial case studies and the results were compared with those yielded by the sequential version. The time savings achieved thanks to the parallelisation were significant. Besides, in the parallel version, more paths can be explored per unit time. In practice, this implies greater robustness.

Keywords: Instrumentation design; Observability; Parallel distributed processing; Networks of workstations

1. Introduction

Observability analysis is a broadly used tool for plant instrumentation design that basically consists in determining which unmeasured variables can be calculated from the measurements by means of model equations. There are two main methodologies to carry out this task, the topology-oriented approach and the equation-oriented approach. The former (Kretsovalis & Mah, 1988) is based on graph theory and applies analysis rules to classify the variables by inspecting a sequence of graphs derived from the process topology. Although efficient, these techniques become less rigorous when the model involves strongly non-linear relationships. As to the equation-oriented philosophy, the structural techniques (Romagnoli & Stephanopoulos, 1980) obtain the classification by means of the structural rearrangement of the model's occurrence matrix. Due to its nature, these methods allow more independence from the degree of non-linearity exhibited by the mathematical model.

Ponzoni, Sánchez and Brignole (1999) developed an equation-oriented structural strategy, called global strategy with first least-connected node (GS-FLCN), which is unique in the sense that it can deal with strongly non-linear mathematical models and proves to be extremely robust. Though GS-FLCN yielded excellent classification results, it became computationally expensive for industrial applications of big size because run-times grew considerably. Therefore, the parallelisation of the algorithm constitutes and efficient way of overcoming this drawback.

Parallel processing is a well-known technique that enables significant reductions in execution time. The traditional approach has been to employ parallel computers. Nevertheless, its applicability is limited by the need for expensive equipment and the lack of manufacturer standards. In contrast, the use of a distributed configuration made up of workstations connected by a local data-communication network allows the efficient use of existing resources, offering minimal start-up budget and easier scalability. In view of these facts, we decided to develop a parallel-distributed implementation for the GS-FLCN strategy, called GS-pFLCN.

* Corresponding author. Tel.: +54-291-4861700; fax: +54-291-4861600.
E-mail address: dybrigno@criba.edu.ar (N.B. Brignole).

0098-1354/01/$ - see front matter
PII: S0098-1354(01)00625-1

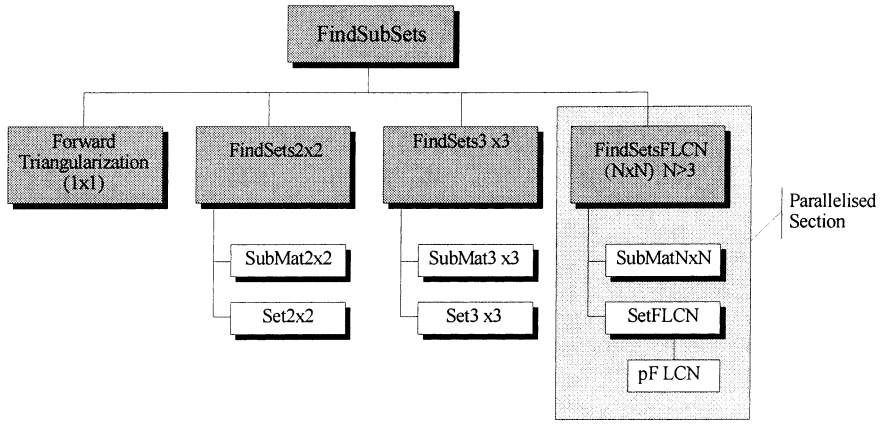

Fig. 1. Hierarchical structure of the GS-pFLCN program.

2. A parallel GS-FLCN formulation

GS-FLCN basically consists in decomposing the process, occurrence matrix **M** by means of an incremental search in order to yield a maximum number of assignment subsets of minimum size. An assignment subset is a block associated to a solvable square subsystem of algebraic equations. Fig. 1 shows the program's hierarchical tree, where the procedure *FindSubSets* guides the search. First, the classical forward triangularisation method detects removable 1×1 blocks. Then, GS-FLCN uses the routines *FindSets*2×2 and *FindSets*3×3 to find subsets of size 2 and 3, respectively. Finally, the first least-connected node (FLCN) algorithm (routine *FindSets*$N \times N$) looks for blocks of order 4 or greater. Each of the subroutines *FindSets*2×2, *FindSets*3×3 and *FindSets*$N \times N$ comprises two main procedures, *SubMat* and *Set*. First of all, *SubMat* builds the input submatrix for *Set*, which carries out the search within it. In particular, *p-FLCN* explores different combinations in parallel from an initial node chosen by *SetFLCN*. Each subtree is explored by pre-ordering from the far left. If no subsets are found, the control is transferred to the nearest subtree on the right, whenever it exists. Otherwise, the program ends. Whenever a subset is found, the control returns to the *Forward-Triangularisation* root to ensure getting the maximum number of blocks of minimum size.

The *FindSetsFLCN* section was the most time-consuming branch in the strategy because it deals with the detection of all assignment subsets of significant size. For this reason, it was convenient to parallelise only this part of the code, while the search for subsets of sizes 1–3 was processed sequentially. The core of this section is a depth-first search (DFS) with heuristics through an undirected graph G corresponding to $\mathbf{M}^{\mathrm{T}}\mathbf{M}$. Each assignment subset can be associated to a path in G, whose nodes correspond to the observable variables. Therefore, the central idea was to devise an efficient way of parallelising a classical DFS algorithm, later incorporating this know-how into a specific parallel FLCN routine.

3. The p-FLCN design

The parallelisation of DFS algorithms has been discussed in the literature (Chaudhuri, 1992). Nevertheless, the classical methods were designed to run on parallel computers and neither their adequacy nor their applicability under distributed environments has been addressed. In particular, Wilkinson and Allen (1999) presented an approach that consists in partitioning the search space into multiple independent DFSs. This implies that breadth-first explorations are triggered from a certain threshold. Each of the subtrees born at the chosen threshold is processed individually as a depth-first search on an available computing node. This strategy was chosen in this work because it can be applied naturally to networks of workstations (NOWs), resulting in a general scheme that corresponds to the well-known master–workers philosophy. The master sends one message to each idle worker that contains information about the subtree to be processed. As soon as a worker finishes its exploration, he returns an end-of-task message with the results.

Fig. 2 shows the general task distribution for *FindSetFLCN*. The choice of the most appropriate

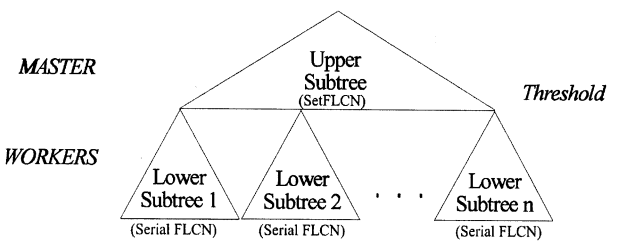

Fig. 2. The tree-search decomposition.

threshold is a key decision that implies a trade-off between search-domain sizes and number of messages. When the threshold augments, moving into deeper levels, the size of the subtrees to be explored by each worker becomes smaller. So, in this sense, a higher threshold would be desirable. Nevertheless, it is clear that the amount of lower subtrees grows exponentially with the threshold, thus leading to an increase in the number of messages. Besides, the amount of work the master carries out sequentially also increases with the threshold. Then, from this point of view, a lower threshold would be better. In view of these opposing trends, we performed some experiments in order to determine the most convenient threshold for this problem. Due to the complexity of the original GS-FLCN code, we implemented an auxiliary DFS program to carry out the tests. Ten representative examples with sparse incidence matrices of order 100 and densities around 5% were employed. The parallel runs for various thresholds and path lengths revealed that the best threshold was one. So, the dominating aspect was the joint effect of the message–passage overhead together with the pure sequential run-time load.

The load-balancing strategy chosen for this implementation was a demand–driven approach, i.e. a dynamic distribution of tasks, assigning them as soon as the workers become idle. This is suitable because the run-times for each task cannot be foreseen. First of all, the densities of the lower subtrees are not equal, thus implying different exploration efforts. Besides, the computational resources are usually shared with other users, thus overloading the processing units dynamically. As a result, the amount of time to be employed by a processor for the execution of each task differs in an unpredictable way during the computations. An additional advantage of a demand–driven approach is that it can also be applied under heterogeneous distributed environments, i.e. where the processors do not have the same performance.

The time complexity (*order*) of an algorithm is expressed as a function of the problem size. It describes the speed with which run-times grow as problem size increases. Given an occurrence matrix **M**, whose dimension is $r \times c$, FLCN's order is:

$$O(\max[r, c](k-1)^s)$$

where k is a constant that depends on **M**'s degree of sparsity and s is the maximum size of the subsets to be explored. The variable k represents the average amount of adjacent nodes in $\mathbf{M}^T\mathbf{M}$, its typical values being real numbers that range between 2 and 4. This formula was obtained for the serial algorithm using recursive recurrence analysis.

For the parallel version p-FLCN, the same amount of work is partitioned and the tasks are distributed among the processors. The time complexity of a parallel algorithm is the sum of the complexity of the computation and the communication. Therefore, if we consider an even task distribution, the computation time complexity for the parallel implementation becomes

$$O\left(\frac{\max[r, c](k-1)^s}{P}\right)$$

where P is the number of processors. In turn, the communication time complexity is $O(c)$.

4. Main results

The GS-pFLCN algorithm was implemented in *C*, employing the PVM message-passing library (Geist, Beguelin, Dongarra, Jiang, Manchek & Sunderam, 1994). Then, the parallel code was applied to classify the unmeasured variables for two real process plants: an ammonia synthesis plant (Bike, 1985) and an ethane plant (Ponzoni et al., 1999). The main concern was to determine whether the existing instrumentation yielded enough information to estimate the values of a set of unmeasured variables of interest. To ensure the accuracy of the results, the analysis was carried out using rigorous non-linear plant models that comprised mass and energy balances.

The size and complexity of the case studies were significant enough to justify the employment of a parallel strategy. The model of the ammonia plant consisted of 560 algebraic equations, starting with 516 unmeasured variables. The ethane plant was represented with 1830 equations and 1425 unmeasured variables. The runs were carried out on a homogeneous environment made up of 10 Pentium 200 MHz workstations with LINUX operating system using a 10 Mb Ethernet local area network. To achieve optimal results on this configuration, the network was isolated and no other processes that could interfere with the runs were triggered during execution. The parallel performance was quantified in comparison with the sequential algorithm by the successive incorporation of processing nodes to the parallel virtual machine. To quantify the benefits derived from the parallelisation, fair comparisons between sequential and parallel computing times were carried out.

Table 1 shows the times elapsed for the sequential and parallel runs. The reductions in execution time were always significant. In this respect, it should be taken into account that the savings are even greater because the parallel algorithm is used repeatedly during the instrumentation design procedure as a whole. In practice, the observability analysis is typically carried out several times before a satisfactory final set of measured variables is reached. During a typical design session, the process engineer proposes a set of measure-

Table 1
Sequential and parallel times (min) for homogeneous distributed processing

	Number of processors (parallel run)					
	1 (Sequential algorithm)	2	4	6	8	10
Ammonia plant	34:19	18:47	09:51	06:58	05:42	04:53
Ethane plant	58:23	36:54	21.54	17:05	14:25	11:53

ments, obtains the final pattern and analyses whether the instrumentation is enough to have full knowledge of the plant. If some of the indeterminable variables are of interest, he adds or removes measurements at the most convenient locations and repeats the procedure.

Figs. 3 and 4 show the performance improvements achieved thanks to the parallel implementation. These comparisons are fair because the reference value for both speed-up and efficiency calculations was the time elapsed when running the sequential version of the same algorithm. The use of the parallel code running on a single processor would have been a poor choice as reference time because the penalty associated with task management would have led to an artificial overestimation of performance values.

Speed-up curves typically have a maximum that indicates a scalability limit, the highest number of processors that is worthwhile adding to the network. In these problems, this critical point has not been reached yet, so more processing nodes could have been incorporated.

4.1. Performance analysis

The parallel performance was always satisfactory in spite of the fact that the runs were carried out on a standard 10 Mb Ethernet network. Better results are to be expected on faster LANs. The theoretical optimum performance corresponds to the upper bound for the speed-up value, known as linear speed-up, which is equal to the number of processors and corresponds to 100% efficiency. This limit represents the ideal situation that arises when the whole algorithm has been parallelised, communication costs are negligible and the processing units have no idle time. For the kind of problems and the hardware architecture addressed in this work, attainable speed-ups are always lower than this bound because the first two conditions mentioned above do not hold.

First of all, the GS-pFLCN algorithm involves a serial section that was not worthwhile parallelising. For the runs reported in Table 1, the elapsed sequential times were 32.45 and 401.37 s for the ammonia and ethane plants, respectively. The former example always exhibited better performance because its sequential part of the code — i.e. the searches for subsets of order lower than 4 — was comparatively smaller. In other words, a higher percentage of the run was processed in parallel. As is clear from Amdhal's law, these values set the limits for the speed improvements that can be achieved by the incorporation of additional processors.

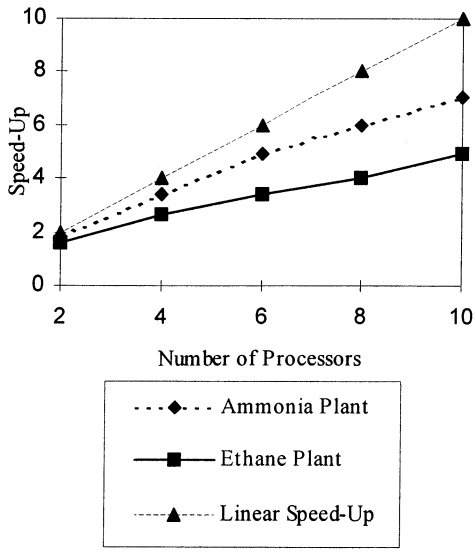

Fig. 3. Scalability speed-up evolution.

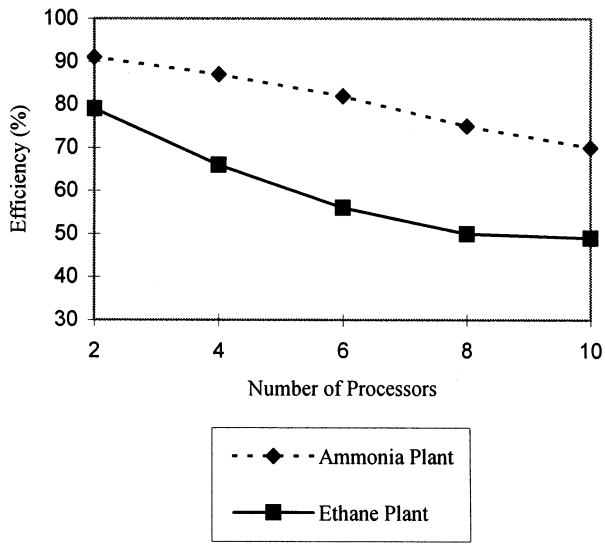

Fig. 4. Scalability efficiency changes.

Table 2
Communication times for the runs reported in Table 1

Number of processors	2	4	6	8	10
Communication times for the ammonia plant (s)	32.15	32.25	32.50	33.10	33.97
Communication times for the ethane plant (s)	101.1	102.01	103.76	105.12	107.3

Besides, communication costs are significant, which is to be expected for a distributed computing environment connected with a slow Ethernet network. Table 2 shows the corresponding values for the two examples presented in this paper. Nevertheless, the impact of communication costs on overall performance becomes lower as the density of the subtrees to be explored in parallel increases.

It is also clear from Table 2 that the degree of network contention is not significant. The increases in communication time due to the addition of processors are very small, which means that there is little interference among the messages passing along the network.

Finally, with respect to idle times, the choice of a demand–driven load–balancing approach is advantageous because the strategy keeps the workers busy most of the time. As a result, there is no load imbalance affecting parallel efficiency.

4.2. Robustness analysis

Since FLCN is combinatorial in nature, assignment subsets of great size are costly to detect. This is clear from the time complexity analysis (see Section 3). In Ponzoni et al. (1999), we had proposed the use of branching factors (BFs), an acceleration strategy that consists in pruning some branches wisely in order to reach deeper levels more quickly. The BFs set upper bounds for k, thus reducing run-times, but this is detrimental to robustness because some potential assignment subsets might eventually be omitted. An additional benefit of the introduction of parallelism into GS-FLCN strategy is the fact that it makes it possible to explore more subtree branches within reasonable times. As a result, less pruning is required, so the risk of missing out blocks is reduced. In practice, this represents a desirable improvement in robustness.

5. Conclusions

The main purpose of this work was to develop a parallel implementation of a novel structural combinatorial technique for observability analysis that rearranges the process occurrence matrix to yield a block lower-triangular form with specific features. In contrast with previous works, the novel procedure, called GS-FLCN, is able to deal with structurally singular matrices satisfactorily. The parallel implementation is highly advantageous for the treatment of realistic instrumentation design problems because significant time savings are achieved in comparison with the sequential approach. Moreover, the parallel methodology is more trustworthy in the sense that higher branching factors can be employed, thus increasing the degree of robustness.

The core algorithm for the parallelisation of this combinatorial technique is a DFS. Therefore, we carried out a preliminary analysis by developing a parallel implementation of a DFS procedure. It was determined that the quality of the performance strongly depends on the threshold choice. In this respect, the study revealed that problems of this kind should be parallelised using low thresholds. On the basis of the DFS results, we developed a parallel master–workers scheme for the observability algorithm and tested it for realistic industrial examples on a network of homogeneous workstations.

An important feature of this implementation is that it is potentially applicable to heterogeneous distributed environments because we adopted a dynamic load–balancing policy. Besides, we employed the PVM library that allows message passage among workstations of different architectures and/or operating systems.

Acknowledgements

We gratefully acknowledge the economic support given by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) through grant PEI 0068/98 and the Agencia Nacional de Promoción Científica y Tecnológica, SeCyT, Argentina, through grant: PICT97 No. 03-00000-01258.

References

Bike, S. (1985). Design of an ammonia synthesis plant. *CACHE case study*, Department of Chemical Engineering, Carnegie Mellon University.

Chaudhuri, P. (1992). *Parallel algorithms*: *design and analysis*. New Jersey: Prentice Hall.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., & Sunderam, V. (1994). *PVM*: *parallel virtual machine. A users guide and tutorial for network parallel computing*. MIT Press.

Kretsovalis, A., & Mah, R. S. H. (1988). Observability and redundancy classification in generalized process networks-I. Theorems.

II. Algorithms. *Computers and Chemical Engineering*, *12*, 671–703.

Ponzoni, I., Sánchez, M. C., & Brignole, N. B. (1999). A new structural algorithm for observability classification. *Industrial and Engineering Chemistry Research*, *38*, 3027–3035.

Romagnoli, J. A., & Stephanopoulos, G. (1980). On the rectification of measurement errors for complex chemical plants. *Computers and Chemical Engineering*, *35*, 1067–1081.

Wilkinson, B., & Allen, M. (1999). *Parallel programming*. New Jersey: Prentice Hall.