An Evolutionary Algorithm to Improve Knowledge-Based Decision-making for Automatic Parallelisation

P. J. P. MCMULLAN and B. MCCOLLUM
School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN
NORTHERN IRELAND

Abstract: - Expert system decision making has proved effective in guiding the process of automatic transformation of sequential legacy codes to parallel equivalents for execution on multiprocessor computer systems. Parallelization decisions normally taken by human experts are replaced by those influenced by domain-specific and 'history' case-based knowledge. However, the information obtained from the knowledge base can be both limited and unreliable, reducing the effectiveness of the decision making process. A time-consuming iterative trial-and-error process is required in order to find the transformation to obtain the best possible performance results. This paper introduces a technique which employs a genetic algorithm to guide the process of performance-related solution-finding, while reducing the penalties inherent with exhaustive testing.

Key-Words: - Automatic, Intelligent, Parallelisation, Distribution, Expert-Systems, Knowledge-Based, Multiprocessor, Evolutionary-Algorithm, Genetic-Algorithm, Case-Based-Reasoning

1 Introduction

The challenge of transforming legacy codes to an equivalent form for execution on parallel computer systems has generated a considerable amount of research [1][2]. The conversion process requires the selection and application of a set of transformations to eliminate dependencies and generate efficient data distributions. The human expert makes transformation decisions to achieve the goal of producing a performance gain of parallel execution over sequential execution. A system which emulates this decision making process is required to replace the human expert. This can allow novice users to produce efficient parallel code. Artificial Intelligent techniques can be used to guide the automatic parallelization process via informed transformational decisions. For example, the choice of an appropriate data partitioning algorithm can highlight the effectiveness of the expert system.

The *KaTT* system [3][4][5] consists of a suite of transformation and code restructuring tools controlled by a sequence of expert system guidance tools. An Input Handler accepts the sequential source code and converts to an intermediate form; a hierarchical graph which describes the syntactical structure of the code. This is then fed to the Transformation stage, which restructures the graph in an attempt to remove or reduce potential obstacles to the production of a parallel solution. The semantic meaning of the original code remains consistent throughout all transformations.

The Generation and Evaluation Stages are then used to convert the transformed graph to a parallel form (known as the potentially parallel graph) and create a set of parallel processes. An Expert system controller is used to influence transformation and data distribution decisions. The system uses a knowledge base built from a number of specific knowledge sources, including platform-specific knowledge, source code characteristics, performance statistics and case-based parallelization strategy records.

2 Knowledge Based decisions

KaTT uses the CLIPS Expert System to utilize the information from the available knowledge sources during each stage of Generation and Evaluation. The expert system initially makes evaluations of the code based on estimation results provided by a Program

Modeler. This guides the system to compute-intensive areas of the code for further analysis. Sequential performance analysis allows the expert system to identify exactly areas of the sequential code upon which to concentrate data partitioning transformations.

A Data Distribution tool performs the main partitioning transformations. Initial decisions for distribution are based on the analysis results previously obtained, such as sequential performance measurements, the occurrence of partitioning obstacles such as data dependencies and target platform considerations. Parallel execution performance results are used by the expert system to control continual improvement attempts and determine when to terminate the improvement cycle.

At the data distribution stage the system has enough information to compile a list of possible distributions for the sequential code. Although there are a limited number of distribution solutions, there are many more optimization techniques which can be applied in an attempt to improve the solution and find the most effective configuration. Therefore, a large list of potential transformation solutions can require testing.

Testing each potential solution requires applying each transformation configuration to the sequential code in order to find the most suitable, i.e. the transformation which provides the best performance results when executed on the target parallel platform. The goal is to find the optimum solution among the list of suitable solutions. A "best-so-far" process is employed to keep a record of the most effective transformation during the improvement cycle.

3 Improvements to the Process

The iterative methods applied within the improvement cycle provide a method for finding the optimal transformation solution to apply to the original sequential code. However, a major disadvantage can be identified.

Given a sequential code with a large number of possible transformations, the problem lies in finding the most effective transformation in a time-frame acceptable to the user of the system. The improvement cycle may require a large number of iterations in order to ensure that the best performance has been found. The expert system provides guidance for the choice of optimization parameter values. However, it cannot dismiss possible solutions in the belief that the performance results will be poor. All potentially peformance-improving techniques must be applied. Testing one solution itself may take some time, therefore obtaining parallel performance statistics for many configurations can be considered unfeasible.

A process is required to reduce the potential volume of testing, while retaining the power to find the optimum solution for the problem. The problem in this context can be defined as:

'*The transformation solution to apply to a given sequential legacy code which will produce a performance speedup when executed on a multiprocessor system*'.

One method is to use the 'experience' gained from a historical record of past parallelization cases [5]. This requires a database of optimum transformation solutions for a wide range of codes previously parallelized (by hand or using iterative methods). Each problem, and the associated most effective solution, are stored in the database along with the performance results obtained. It then may be possible to find a matching problem within the database, and use the previously recorded transformation. The stored performance results indicate the potential parallel execution performance which could be obtained by applying this transformation to the sequential code.

The database effectively becomes a knowledge base to supply satisfactory transformations quickly without the need to test for all possibilities. The pattern matching capabilities of modern Expert System shells make the creation of such a knowledge base within the system a possibility.

This method is effective for problem cases to which an associated solution exists within the Knowledge Base. However, there are further issues to address regarding the operation and effectiveness of this method. A knowledge base with a limited amount of cases may not contain any codes similar to the code under consideration and make this method unproductive. Furthermore, due to the complexity of structural programming languages, codes which are similar may exist in the knowledge base, but may not be similar enough to guarantee the effectiveness of the associated solution when applied to the sequential code.

Applying a degree of 'fuzziness' to the technique will enable matches to be made based on relevance to the original problem. If the current case matches a

case in the knowledge base almost exactly, the relevance is high. As the degree of similarity between cases decreases, the measure of relevance decreases accordingly.

A reasonable solution can be found, although a low relevance factor can undermine the performance potential associated with the solution. Improvement is required to determine if this solution is indeed the most effective possible. An iterative improvement process may be required, thus removing any benefits gained from using the knowledge base technique.

Finding a relevance level which guarantees a single satisfactory match can be difficult. Depending on the size of the knowledge base, a range of possible matches can be returned. Given an effective combination of improvement, any of these transformation solutions could potentially yield the best possible performance results. An alternative to the iterative method is required to explore this area without incurring any time-consuming side effects.

An improvement to this process has been attempted by employing *Case-Based Reasoning* techniques for the selection of potential solutions and the application of an Evolutionary Algorithm to control the progress of applied improvements. Research is concentrated in the main areas of:

1. Effective Case retrieval
2. Solution transformation or improvement using an Evolutionary Algorithm
3. Case Storing, to improve future searches.

Various models of case retrievers such as ARCHIE [6] and DEJAVU [7] are used within the engineering design domain. The most appropriate method for the parallelization problem domain is that adopted in the framework of the generic tool CASETOOL [8].

4 Case-Based Selection

During the Case-Based Reasoning process, searching is guided by previous problem-solving experience. These are stored as past cases, enabling solutions to the problems already solved to be retrieved rather than computed again. Within the problem domain of parallelization, cases contain characteristics which define the code. The associated transformation solution is stored along with performance results obtained during execution of the transformed parallel program.

A generic methodology for Case retrieval involving the following three steps is applied to this problem-domain:

1. Selection by Search Conditions: Selecting a set of cases after removing all loosely connected cases.
2. Classification by Relevance: Classifying cases based on the degree of similarity of characteristics between the given situation and the selected cases.
3. Classification by Performance: Classifying cases based on the past performance as a prediction mechanism for potential performance results for the given situation.

The relevance of selected cases is based on the deviation of attribute values of cases from that of the current selection and the relative importance (weighting) of the attributes. The performance of cases with a higher relevance are “trusted” to a greater degree than those with a lesser relevance.

A *Relevance Norm (R)* is evaluated to represent the similarity between past cases and the present problem situation. The value *R* is the normalized weighted least square estimation of deviations [9]. The evaluation of *R* can be illustrated using the equation in Figure 1:

$$R = \frac{1}{n} \sqrt{\sum_{i=1}^{n} w_i \left(\frac{v_i - c_i}{u_i - l_i} \right)^2}$$

Fig. 1 Fitness Function

where $v_1 .. v_n$ is the set of n attribute values of the past case, $c_1 .. c_n$ the corresponding values of the current situation and $w_1 .. w_n$ be the weights of these attributes. The variables u_i and l_i are the upper and lower limits for the i^{th} variable. The classification for values of *R* is domain dependent, and is determined by the Expert System base on in-built parallelization strategy rules. Cases which fall into reasonable system-defined classifications such as *perfect* or *close* can be considered for selection.

The characteristics of common compute-intensive areas of program code such as loops are stored in the knowledge base. Typical attributes or characteristics for loop cases describe the loop itself and statements

within the loop body. The simplest type of loop *(1 dimensional, with literally declared bounds)* can be described as:

`1-dimensional Loop (`*`lb/ub`*`)`

where *lb* and *ub* are the upper and lower bounds.

More complex *(n-dimensional)* loops can also be represented in the knowledge base, along with specific information such as loop body statement patterns for each level of nesting. For complex cases, relevance is a crucial factor in decisions regarding accuracy of matches. The performance of a selected case can suggest the potential performance of the current problem if the associated case-solution transformation has been applied. The relevance however, can affect the reliability of this solution and therefore that of achieving the potential performance.

It may be more rational to select a case with a higher relevance factor over one with lower relevance factor, even if the latter has a higher performance potential. Determining the 'fitness' of one solution over another solution can help allow the system to make this choice. Fitness can be described as a function of the relevance of the selected case and the performance of the associated solution.

Given the likelihood that a perfect match will not be retrieved from the knowledge base, the system must be capable of choosing between a number of possibilities, each with varying degrees of reliability. Simply choosing the fittest from a list of candidates may not ultimately yield the best results, for a number of reasons, e.g.

1. The fittest solution may still require improvement.
2. Improvements suggested by the expert system may be limited and/or misguided.

Improvements performed on a less healthy solution may yield a better final result than that achieved by improving on the fittest. The use of an evolutionary algorithm enables a more explorative search to be performed.

5 An Evolutionary Algorithm

Evolutionary techniques involve the use of computer-based solving systems which use computational models of evolutionary processes as key elements in their design and implementation. The term Evolutionary Algorithm describes an iterative and stochastic process operating on a set of potential solutions to a given problem (known as the population). Such algorithms are applied to diverse application areas [10]. *Genetic Algorithms* [11] fall under the classification of Evolutionary Algorithms. They combine Darwinian theory of survival-of-the-fittest and natural genetics to form a robust search mechanism. The adaptive nature of a genetic search simulates learning from the problem environment as the search progresses. Such learning guides the search technique to arrive at global optimal solutions.

Genetic Algorithms and Expert Systems have been effectively combined with numerical optimization techniques in a process called *Interdigitation* [12]. The genetic Algorithm is employed for global search while the expert system handles local searches for engineering design optimization problems. Hamada et al. [13] has developed a hybrid model for Genetic Algorithms and Knowledge-Based Systems for production planning in steel plants.

5.1 Introducing the Genetic Algorithm

The steps in the algorithm are illustrated in Figure 2:

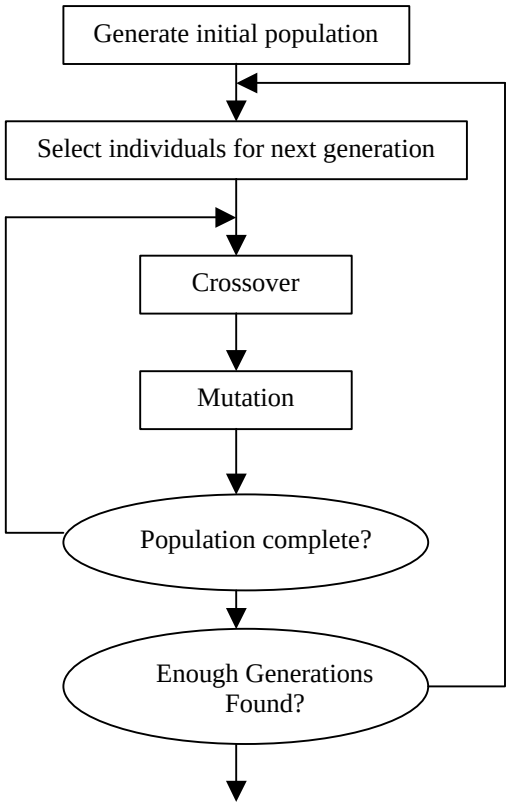

Fig. 2 The genetic algorithm

The fitness function for each selected case in the knowledge base is used initially to focus on the most potentially suitable candidates for improvement (the initial population). At the initial selection stage, a temporary population is created, containing the fittest individuals. Reproductive operators are applied to this population, and a new population is created. Finally, individuals of the original population are substituted by the newly created individuals. This replacement tries to keep the best individuals and keep the population size constant. The process is repeated until a certain termination criterion is achieved (usually after a given number of iterations).

The reproductive operators are applied in an attempt to improve on the current population while retaining the best characteristics of the fittest individuals.

5.2 Applying this to Characteristics

A transformation Solution contains the necessary information to guide the expert system and the data distribution tools to create the required parallel version of the sequential code. At the most basic level, this information will include *Number of Processors to Use* and *Type of Data Distribution to Apply*. Communication and Processor optimization parameters are also applied during transformation.

This leads to many slightly different possible transformations for each processor/distribution combination. There may be many candidates for initial selection, each with an associated fitness. For a required initial population of *n* solutions, picking the best *n* (those with highest fitness) from the candidate list may not be the best initialisation strategy.

Maintaining a level of diversity is important as the best solutions may be quite similar to each other. A limited set of potential improvements can then be produced during each generation of the new population. *Tournament selection* is used for initial seeding. This picks the best individuals from a randomly generated subset.

The selected individuals undergo a set of transformations as a result of reproductive operators. Crossover, or recombination is the first operator to be applied. One technique is *two-parent crossover*, in which Parent individuals are selected for combination, and offspring containing elements of both are produced. Mutation is then applied to the newly created offspring population in order to diversify the new population. The amount of mutation depends on the fitness of the parents.

At this stage the improved individuals must be re-evaluated for fitness. The system cannot rely on previously stored information on performance of the new solution, so parallel profiling techniques are used. A new set of individuals are then selected to continue the process. The Termination criteria determine when to stop creating new generations, e.g. stop after a set number of generations.

6 Future Work

Initial results have been encouraging, using a limited knowledge base for this method. However, a full set of results using a larger knowledge base is currently being compiled and will be published in due course. These results will also be used to compare to the original KaTT system under the Expert System driven iterative search method.

The Expert system guidance has proven effective in eventually obtaining a program transformation to obtain the best possible performance results. Work is also on-going to implement this guidance within the actual Genetic Algorithm to help in selection, population reproductive and replacement decisions. This knowledge will be obtained through rigorous use of the system under various methods given differing circumstances. For example, the tournament method of initial selection may be effective for some problems, but an alternative method such as *bias selection* may be more effective for others.

Maintaining diversity is an issue which has warranted considerable research [14]. During the initial selection process, maintaining diversity over the problem space can be problematic. The expert system will attempt to ensure diversity is maintained at this stage and further subsequent selections.

The results for each successful implementation of the evolutionary technique are stored in the knowledge base. Along with this, further information is stored regarding the steps taken within the process. This information can be used to influence decisions during the Case selection process. For instance, the fitness of a stored case can be further influenced by the effectiveness of the Genetic Algorithm process to obtain the associated solution. Further environment-base characteristics will also be introduced to influence the fitness of a solution. For example, different hardware platforms or processor topologies can produce different performance results for the same parallel transformation. Heterogeneous clusters exhibit varying processor speeds and communication

or message-passing requirements not relevant to High Performance multiprocessor systems.

Memetic Algorithms [15] are an extension of Genetic Algorithms to include local search mechanisms such as the *Hill-Climbing algorithm*. Applied to the problem-domain of parallel transformations, the mutation operators will be heavier, giving a higher possibility of unfeasible solutions. The local search will improve the mutated solutions with the intention of arriving at the optimal solution much quicker than with standard Genetic Algorithm. Investigation is based on whether the introduction of local search heuristics may improve the process for this problem-domain.

7 Conclusion

The emulation of human expertise within the problem of automatic parallelization of legacy code requires the production of an efficient parallel code yielding a satisfactory performance improvement. The cost of time taken to produce this improvement must influence the measure of the effectiveness of a system undertaking this task. Current work is involved in reducing the time taken to produce efficient parallel versions of the sequential legacy codes, while maintaining or improving the quality of the final parallel code. Initial results show that improvements are possible. It is contemplated that combination of the expert system and expansion of current evolutionary techniques will further improve the solution-finding process in the current system.

References:

[1] B. Chapman, T. Fahringer and H. Zima, Automatic support for data distribution on distributed memory multiprocessor systems, in U. Banerjee et al. Eds. *Proceedings of the 6th Workshop in Language and Compilers for Parallel Computing*, New York: Springer-Verlag pp184-199, 1993

[2] P. F. Leggett, A. T. J. Marsh, S. P. Johnston and M. Cross, Integrating User Knowledge with Information from Parallelisation Tools to Facilitate the Automatic Generation of Efficient Parallel Fortran Code, *Parallel Computing*, vol. 22, pp259 - 288, 1996.

[3] P. Milligan, P. P. Sage, P. J. P. McMullan and P. H. Corr. A Knowledge Based Approach to Parallel Software Engineering. In, *Software Engineering for Parallel and Distributed Systems*, Chapman and Hall, ISBN 0-412-75640-0, pp 297 - 302, 1996.

[4] P. J. P. McMullan, P. Milligan and P. H. Corr, Data Distribution, Analysis and Evaluation of Code - An Expert System Approach, *Proceedings of the IEEE/IMACS CSCC '99*, 3rd World Multiconference on Circuits, Systems, Communications and Computers, 1999.

[5] McCollum B.G.C., Milligan P. and Corr P.H., "The Structure and Exploitation of Available Knowledge for Enhanced Data Distribution in a Parallel Environment", IMACS/IEEE CSCC'99, ISBN 960-8052-00-9, pp 3301-3307, Athens, 1999.

[6] A. Goel, J. Kolodner, M. Pearce and R. Billington, Towards a case-based tool for aiding conceptual design problem solving, *in Proc. Of the DARPA Workshop on Case-Based Reasoning*, Pensacola Beach, 1989.

[7] T. Bardasz, DEJAVU: Case-based reasoning for mechanical design, *AIEDAM*, 1993.

[8] H. Shiva Kumar and C. S. Krishnamoorthy, A frame-work for case-based reasoning in engineering design, *AIEDAM*, 9, 1995.

[9] P. L. Bergan and R. Clough, Convergence criteria for iterative processes, *AIAA*, 1978.

[10] Proceedings of the Genetic Evolutionary Computation Conference, GEKKO, 2001.

[11] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, *Addison-Wesley*, 1989.

[12] D. Powell, M. Skolnick and S. Tong, *Interdigitation*: A hybrid technique for engineering design optimization employing Genetic Algorithms, expert systems and numerical optimization, *Handbook of Genetic Algorithms*, L. Davis (Ed.), Van Nostrand Reinhold, New York, 1991.

[13] K. Hamada, T. Baba, K. Suto and M. Yufu, Hybridizing a genetic algorithm with rule-based reasoning for production planning, *IEEE Expert*, October 1995.

[14] E. Burke, J. P. Newall and R. F. Weare, Initialization strategies and diversity in evolutionary timetabling, *Evolutionary Computation* 6(1), M.I.T., 1998.

[15] P. Moscató, Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, *Caltech Concurrent Computation Program*, 1989.