An Improved Genetic Algorithm for Task Allocation in Distributed Embedded Systems

Allan Tengg
Institute for Technical Informatics
Graz University of Technology
A-8010 Graz, Austria
tengg@iti.tugraz.at

Andreas Klausner
Institute for Technical Informatics
Graz University of Technology
A-8010 Graz, Austria
klausner@iti.tugraz.at

Bernhard Rinner
Institute of Networked and Embedded Systems
Klagenfurt University
A-9020 Klagenfurt, Austria
b.rinner@computer.org

Categories and Subject Descriptors: D.2.11 [Software Architectures]: Domain-specific architectures

General Terms: Algorithms

Keywords: Genetic algorithms, task allocation problem

1. SUMMARY

The main idea of our I-SENSE research project [1] is to provide a generic architecture which supports online data fusion on a distributed embedded system. To accomplish high flexibility, we decided that the functional description of a data fusion system, the so called *fusion model*, should be defined independently of the present hardware configuration,the so called *hardware model*. It is the objective, to automatically find a valid mapping of the *fusion model* to this *hardware model*. Currently not many publications can be found that report the successful utilization of GP to solve such task allocation problems. Here we present our task allocation method that utilizes GP to find a suitable solution in an adequate time for this optimization problem known to be NP-complete. To improve the performance of the algorithm significantly for our problem domain, we added a few heuristics which are outlined in the following.

The *fusion model* consists basically of a set of communicating tasks which may be represented as a task graph $G = (N, E)$. It is assumed to be a weighted directed acyclic graph, consisting of nodes $N = (n_1, n_2, ..., n_m)$ which represent the fusion tasks and the edges $E = (e_{12}, e_{13}, ..., e_{nm})$ the data flow between those tasks. Each node has some properties, describing the hardware requirements of a task. Every edge indicates the required communication bandwidth between two tasks.

The *hardware model* describes the heterogenous distributed embedded system, consisting of different processors with different properties such as core type, clock frequency and available memory, where the fusion application should run on. Each hardware node has at least one general purpose CPU and optionally some digital signal processors (DSPs) coupled strongly via PCI and I/O ports to connect sensors. Ethernet is used to interconnect the hardware nodes.

The traditional genetic algorithm suffers a major drawback if it is applied to our problem: The percentage of valid combinations is usually very small compared to all possible task allocations. When creating a pure random initial population, it is highly likely that there is not a single valid chromosome in the population. As proposed by Jens Gottlieb [2], a penalty in the fitness score for invalid chromosomes has proven to be a good approach for such situations.

We found further, that the straight forward implementation of the GA with penalties performs worse with increasing problem size, if there is no correspondence of the genes and genetic operations in the real world problem. Without precaution neither the order of the genes, nor the point of intersection in the crossover operator has a meaning in the task allocation problem. This leads to a unrestricted search in the entire solution space with the disadvantage that many invalid allocations have to be checked and ruled out.

We propose, that tightly coupled tasks should be mapped on adjacent genes in the chromosome. Now the genetic crossover operator has a real world meaning: Rather than individual tasks without relation among each other, entire coupled clusters of tasks are exchanged by the crossover operator which leads to a better performance. If the clustering idea is introduced to the initial population as well, a further enhancement can be accomplished. Instead of placing the tasks in the order they appear in the *fusion model*, tasks with high communication requirement are placed first at a random processor together with their heavy coupled tasks. This results in a relatively fit initial population and therefore faster search, compared to a pure random initial population (cp. table 1).

Tasks	CPUs	Complexity	Normal	Enhanced
15	6	easy	2.83	1.84
26	10	medium	13.93	6.79
20	14	hard	81.20	24.00
26	14	hard	58.40	38.93
20	12	very hard	203.86	25.79

Table 1: Iterations to find a optimal configuration

2. REFERENCES

[1] A. Klausner, B. Rinner, and A. Tengg. I-SENSE: Intelligent embedded multi-sensor fusion. In *Proceedings of the 4th IEEE International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, Austria, June 2006.

[2] J. Gottlieb. On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. In E. J. W. Boers, S. Cagnoni, J. Gottlieb, E. Hart, P. L. Lanzi, G. R. Raidl, R. E. Smith, and H. Tijink, editors, *Applications of evolutionary Computing: Proc. EvoWorkshops 2001*, Berlin, 2001.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
ACM 978-1-59593-697-4/07/0007.