A Genetic Algorithm-Based Task Scheduler for Parallel Processor Systems with Non-negligible Inter-Processor Delay

Gregory R. Kramer and Frank W. Moore
Miami University
Systems Analysis Department
230 Kreger Hall, Oxford, OH 45056
{kramergr, moorefw}@muohio.edu

Abstract

The multiprocessor task scheduling problem is an NP-complete problem that is difficult to solve via traditional methods. Recent investigations have used evolutionary computation to evolve optimized schedules, but have failed to address the effects of communications delay on the schedules. As such, the schedules produced are often less than optimal. The goal of this project was to extend previous research by developing a genetic algorithm that can evolve optimized solutions to complex task scheduling problems for multiple processor systems, while taking into account the non-negligible communications delay that exists between processors.

1. Background

The multiprocessor scheduling problem is generally stated this way: given a multiprocessor computing system and numerous tasks to execute, how does one efficiently schedule the tasks to make optimal use of the computing resources? In general, a deterministic search of the solution space to identify an optimal solution to this NP-complete problem is computationally and temporally exhaustive. The problem difficulty depends chiefly upon the following factors: the number of tasks, execution time of the tasks, precedence of the tasks, topology of the representative task graph, number of processors and their uniformity, inter-task communication, and performance criteria. Classic solutions to this problem use a combination of search techniques and heuristics. While these techniques often produce adequate solutions, the resulting schedule is usually suboptimal. These techniques are also criticized for lacking both scalability and performance guarantees.

(Bohler, Moore, and Pan 1998) described the implementation of a genetic algorithm for minimizing the schedule length of a task graph to be executed on a multiprocessor system, and identified several improvements over state-of-the-art solutions. Their paper defined the multiprocessor scheduling problem as a parallel program represented by an acyclic directed task graph. Precedence relations were derived from the task graph, and the execution time of each task was randomly set. All processors were assumed to be identical, were non-preemptive, and used the shared memory model (zero communication delays between tasks). Unlike previous solutions (e.g., Hou, Ansari, and Ren 1994), their program was scalable and adaptable to a variety of task graphs and parallel processing systems. The number of tasks was easily changed to allow schedule optimization for a variety of task graphs with various numbers of tasks. Task and task graph characteristics such as execution times and precedence were readily modified. The number of processors on the target multiprocessor system was freely modifiable to yield schedules for arbitrary parallel processors.

2. Implementation

The assumption of zero communication delay was the primary deficiency of Bohler *et al*'s algorithm. The purpose of this research was to extend Bohler *et al*'s algorithm to account for non-negligible interprocessor communication. The main changes were to the fitness evaluation function and the addition of a processor delay file that shows the time it takes for a source processor to communicate with a destination processor. This file, named delay.dat, must reside in the same directory as the x_tasks.dat and x_edges.dat files. When evaluating the

fitness of an individual, the fitness function previously stated a task's start time as the latest of the finish times of its predecessors. Currently the fitness function defines a task's start time as the latest of the finish times of the task's predecessors plus the time it takes to communicate the predecessor's finished state to the processor on which the task in question lies. This minor addition to the fitness algorithm produces much more efficient schedules than the previous implementation, especially when the processor communication delay time is quite high, as it may be over a LAN between two processors that are working in parallel to solve a problem.

3. Test Results

To test the effectiveness of the new implementation, five runs were completed and the results were compared to those from the Bohler *et al* implementation. In each case, the execution times of the tasks and their dependencies were randomly assigned, as were the delay times between processors. The results of each run and the summary of all the trials are displayed in Figure 1.

Version 1 represents the unmodified schedule program that assigns tasks to processors without regard to inter-processor communication delay. Version 2 represents the modified scheduler that factors in communication cost when creating task schedules. For each of the five test runs, 300 generations were produced, each with 40 genomes per generation.

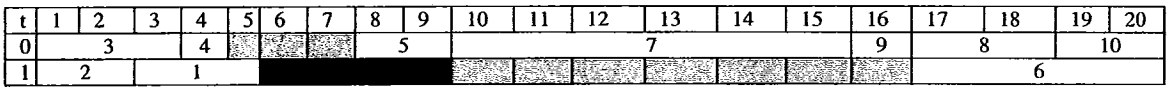
The results were dramatic. In each run, the new version of the program outperformed the previous version and created a faster schedule. The average improvement was a schedule that ran 18.5 percent faster than that of the previous version. However, one of the schedules (Trial #4) was a 50 percent improvement. This is a significant improvement in performance that can be directly attributed to the consideration of interprocessor communication delay by the fitness function during schedule evolution.

Fig. 2 illustrates the final schedules generated during Trial 1. Fig. 3 shows the task graph for Trial 5, while Fig. 4 shows the corresponding final schedules. Numbered cells are scheduled tasks and the cell length represents task execution time. Black cells are unused processor time and light gray cells represent communication delay that keeps a task from starting. Total schedule time is the time of the last block in the chart.

Trial #	Task Count	Processor Count	Version 1 Best Time	Version 2 Best Time	Improvement
1	10	2	21	20	4.7%
2	20	2	45	42	6.6%
3	20	4	33	26	21.2%
4	20	8	54	27	50%
5	40	8	59	53	10.1%

Fig. 1. Results for various task and processor counts.

Version 2 results – WITH INTERPROCESSOR COMMUNICATION DELAY



Version 1 results – NO DELAY

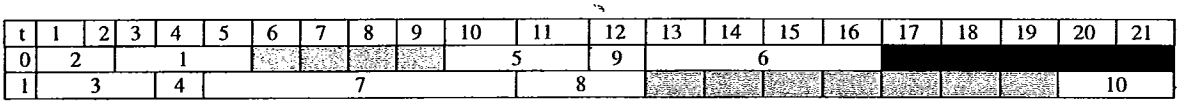


Fig. 2. Trial #1 – Optimized schedules for 10 tasks, 2 processors.

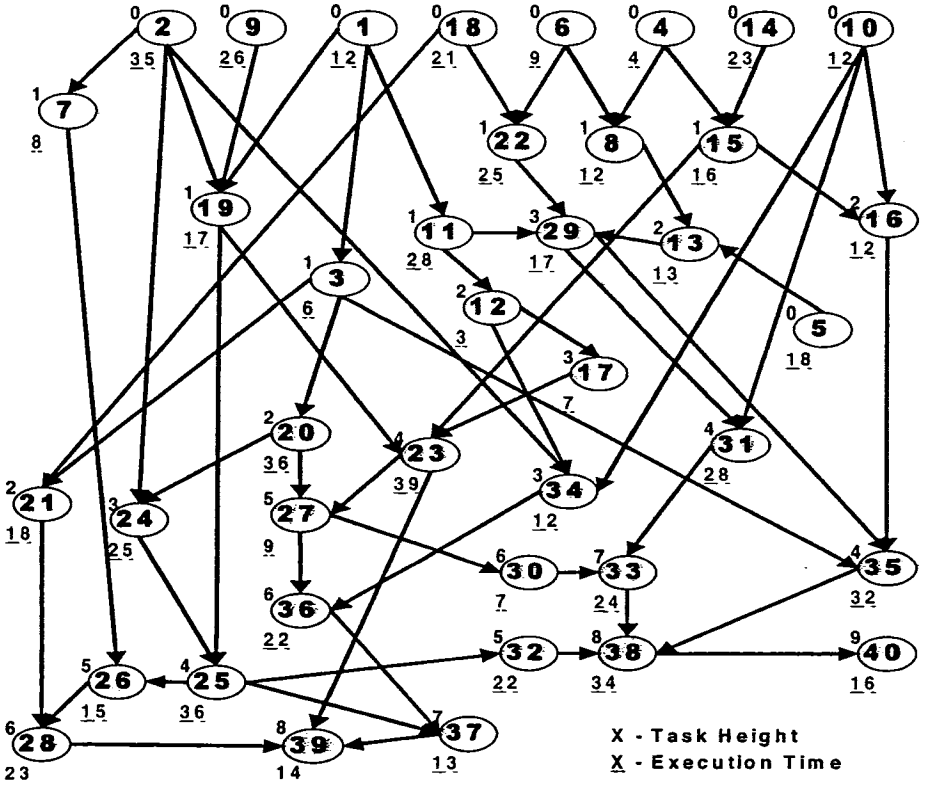


Fig. 3. Task graph for Trial 5 (40 tasks, 8 processors).

40 tasks 8 processors – with delay

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	32			11			34											37	
1	21					36	35		19	26		25			29				
2	24						10										33		
3	22						12							40	14		18		
4	6						30			3			7						
5	8																		
6	23									28									
7	1					16				17			5					27	

20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
37						2											
29																	
33																	
18								38						39			
7														31			
27				15					13		4			9			20

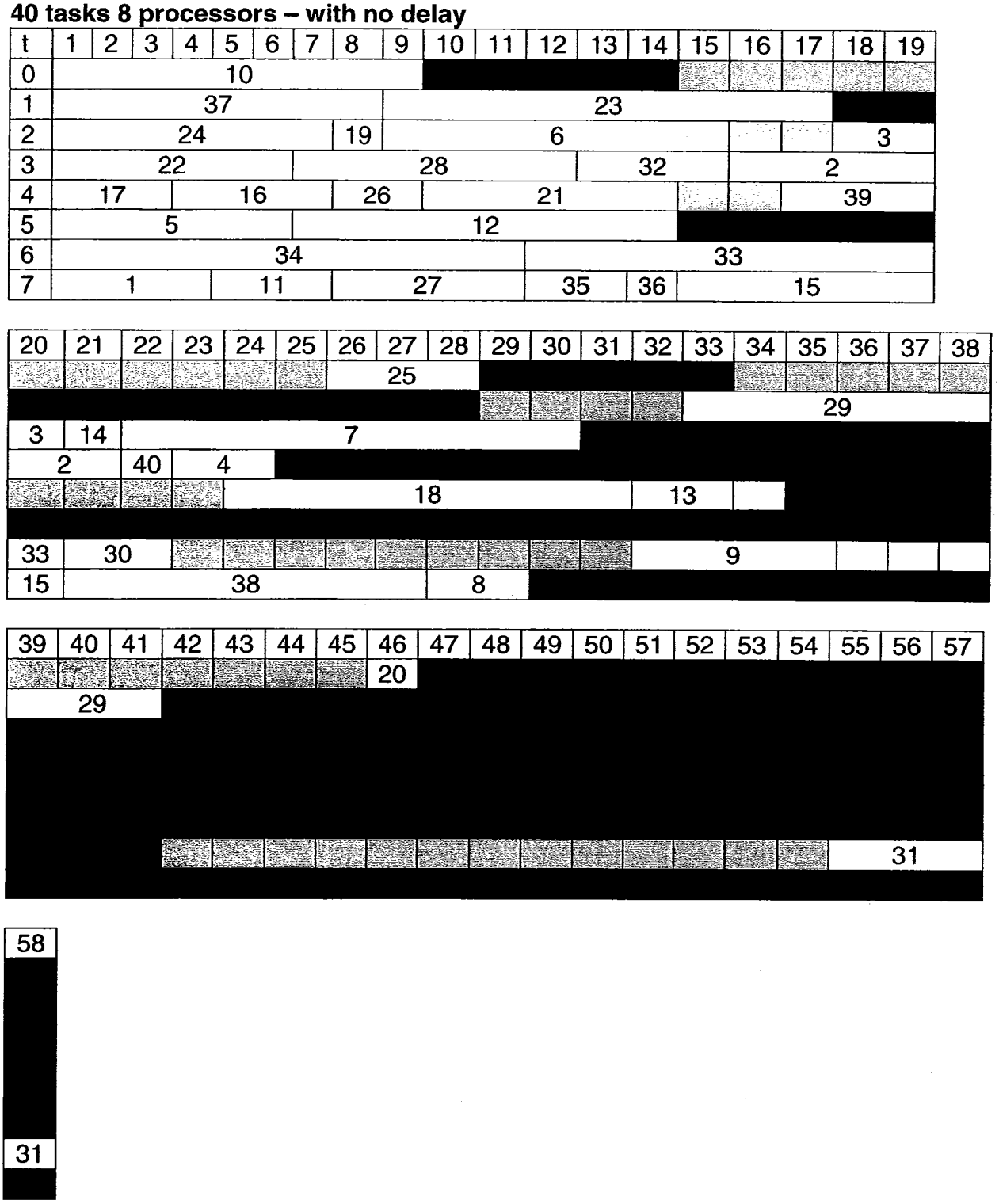


Fig. 4. Trial #5 – Optimized schedules for 40 tasks, 8 processors.

4. Future Work

Even with the significant improvement in performance, there is still much work to be done to increase the usability and scalability of the program. One of the most important future enhancements will be to completely rewrite the program with a GUI interface so that the user may easily reconfigure the program without recompiling the source for every run. Another change will be to rewrite the program as a multi-threaded application, so that the program may take advantage of multiple processor machines in order to schedule arbitrarily complex task graphs involving hundreds of tasks and many processors.

Bibliography

Bohler, M., F. W. Moore, and Y. Pan, 1999. "Improved Multiprocessor Task Scheduling Using Genetic Algorithms", in Kumar, A. N. and I. Russell (eds.), *Proceedings of the Twelfth International Florida AI Research Society Conference (FLAIRS-99), Orlando, FL, May 3-5, 1999*, pp. 140-146, AAAI Press.

Hou, E. S. H., N. Ansari and H. Ren 1994. "A Genetic Algorithm for Multiprocessor Scheduling", in *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5 No. 2, pp. 113-120, Feb. 1994.