Comparison of nature inspired and deterministic scheduling heuristics considering optimal schedules

Udo Hönig and Wolfram Schiffmann
FernUniversität Hagen
Lehrgebiet Rechnerarchitektur
58084 Hagen, Germany
E-mail: {Udo.Hoenig, Wolfram.Schiffmann }@FernUni-Hagen.de

Abstract

We report about a performance evaluation of nature inspired stochastic vs. conventional deterministic scheduling algorithms. By means of a comprehensive test bench, that comprises task graphs with diverse properties, we determined the *absolute* performance of those algorithms with respect to the optimal solutions. Surprisingly, the nature inspired stochastic algorithms outperformed all the investigated deterministic algorithms.

1 Introduction

The analysis and comparison of scheduling heuristics is subject of many recent publications. The usually used test sets consist either of randomly generated task graphs or program traces of mathematical algorithms (e.g. Gauss-Seidel, Cholesky, ...). In very few cases, comparisons to optimal schedules were conducted. Although many authors use big task graphs to evaluate their heuristics, the number of test cases is rather small.

Unfortunately, most authors do not reveal their test sets, making it impossible for other scientists to compare their own algorithm to the analyzed ones. For our best knowledge, Kwok and Ahmad [1] are the only authors who published the used test set on their web-page. It consists of 350 task graphs of different sizes and is structured according to the method the task graphs were produced and the Computation to Communication Ratio (CCR) which represents the ratio of a task graph's node- and edge-weights. Since some of the test case classes consist of only five task graphs, it is possible that a single outlier could distort the analysis' results.

For these reasons, we developed a comprehensive structured test bench with 36000 test cases [2]. The task graphs were generated randomly and structured concerning the graph's size, its meshing degree, its average edge length and the node- and edge-weights. To emphasize a certain graph property (e.g. a high meshing degree), the random numbers were determined by a Gaussian distribution. Since this test bench should also provide test cases which are unbiased with respect to one or more graph attributes, test classes with uniform distributed attributes are provided as well.

2 Survey of the investigated algorithms

We used the current interim version[1] of this test bench to compare some well known scheduling algorithms, namely Dynamic Level Scheduling (DLS), Earliest Time First (ETF) and Modified Critical Path (MCP) with implementations of three nature inspired heuristics, Ant Colony Optimization Algorithm (ACO), Genetic Algorithm (GA) and Simulated Annealing (SA). Since Kwok and Ahmad [3] provide a detailed description of the observed deterministic algorithms, this section concentrates on some aspects of our nature inspired heuristics' implementations. An overview of these meta-heuristics' general properties is given by Blum and Roli [4]. Since nature inspired heuristics have many parameters for tuning their processing, their results can only represent a single implementation and its parameter settings but not the overall meta-heuristics it is based on.

To provide a fair comparison between the nature inspired and the deterministic heuristics, we decided not to make use of a heuristic for the determination of a starting point for the nature inspired algorithms. Otherwise it would be possible to select the best result of all the deterministic algorithms, implying that the nature inspired heuristics would perform better for every situation.

The process of scheduling a task graph can be subdivided into two phases: the selection of the task which will be mapped next and the mapping of this task to an idle processor. Since all of the observed deterministic heuristics perform a greedy mapping[2], we decided to reduce the complexity of the nature inspired heuristics' search spaces by reducing their search to the selection process, too. Therefore, the nature inspired heuristics generate task sequences which are forwarded to a

[1] Since the computation is still in progress, only 30511 optimal solution are currently available.

[2] This means, that a task is mapped to the processor where it can start as soon as possible.

greedy mapping mechanism that allocates the tasks to the processors. The resulting schedules are returned to the heuristics for evaluation and optimization of the search process.

Each individual of the genetic algorithm's population is a valid task sequence. Its fitness is computed by mapping the tasks in the given sequence to the considered target architecture. To keep the algorithm's runtime low, every run consists of only 36 generations with 32 individuals. While testing this algorithm, these settings, combined with a cross-over rate of 90% and a mutation rate of 10%, achieved the best results.

The starting temperature of the SA's cooling process is chosen with respect to the given task graph problem. The initial task order is generated randomly. To reduce the probability of getting stuck in local optima in the late phase of the cooling process, SA uses a proportional cooling strategy. A task sequence's neighborhood is defined by all sequences where exactly one task is placed at another position.

While SA and GA operate on complete task orders, ACO's virtual ants evaluate pairs of preceding and succeeding tasks. Every ant passes all tasks in a valid sequence and forwards this task sequence to the greedy mapper which returns the corresponding schedule's length. According to this schedule length a certain amount of pheromone is distributed equally to all predecessor/successor pairs belonging to the ant's path. Succeeding ants can use these information to select their path through the available tasks.

3 Results

The comprehensive structure of the test bench described above as well as the availability of optimal solutions is the base for a more thorough analysis than had ever been performed before. To point out the limits of the hitherto used methods, we will start this analysis by comparing the heuristics relatively to each other and to the best solution found by one of them. Next, we will show the additional information that can be achieved by using our comprehensive test bench: Firstly, by using the optimal schedules, the algorithms can be scored absolutely. In this way, one gets a more precise view of the algorithm's real performance. Secondly, the test bench's structure allows a more detailed examination of the observed heuristic's strengths and weaknesses.

The heuristics were pairwise compared considering the test bench's 36000 task graph problems[3]. The results are presented in table 1, where every cell contains the comparisons of the two heuristics which are assigned to the cell's column and row. The uppermost value represents the number of test cases, where the heuristics' results differ. The second value relates to the number of test cases, where the heuristic which belongs to the cell's row finds worse schedule lengths than the column's heuristic. The third value describes the inverse case.

Table 1: Relative comparison of the observed heuristics.

	SA	MCP	GA	ETF	DLS
ACO	1721 932 789	19522 16 19506	4576 367 4209	20614 39 20575	19220 15 19205
DLS	19232 19213 19	13447 6632 6815	19190 18377 813	9643 3539 6104	
ETF	20631 20604 27	16867 9523 7344	20505 19795 710		
GA	4543 4221 322	19425 853 18572			
MCP	19525 19509 16				

Obviously, the nature inspired algorithms find clearly better schedules than the deterministic heuristics. While SA and ACO are by far the most successful of the investigated algorithms, the GA performs still better than the deterministic algorithms from which DLS is the best, followed by MCP and ETF.

The degradation from the best known solution (degFromBest), is defined by means of the found schedule length SL and the best known schedule length SL_{best} as $degFromBest = 100 * \frac{SL-SL_{best}}{SL_{best}}$. This is another characteristic parameter which is widely used in scheduling literature [5]. A high degradation value indicates a strong deviation from the best solution and therefore poor results.

Table 2 shows the observed heuristics' degradation from the best found solution[4], averaged over all 36000 test cases. Again, SA and ACO perform much better than GA. The best deterministic algorithm is DLS, followed by MCP and ETF. Although this comparison gives a very clear view of the algorithms' results' relative quality, it has three significant disadvantages. Firstly, it still does not give any clue about the absolute quality of a heuristic's results. Secondly, since all comparisons are related to the best known solution, this analysis's results is strongly dependent of the heuristics selected for analysis. Choosing another set of algorithms could change the

[3]Note that this comparison is independent from the optimal solutions.

[4]This best solution was found by one of the observed heuristics and might therefore differ from the optimal solution.

results to a large extend. Thirdly, every time a heuristic is added for comparison, the whole analysis has to be repeated.

Table 2: Average degradation from the **best** solution (in %).

SA	MCP	GA	ETF	DLS	ACO
0,048	3,472	0,423	3,738	3,415	0,054

These disadvantages are eliminated by the knowledge of optimal schedule lengths. Table 3 shows the heuristic's average degradation from the optimal solutions of 30511 task graph problems. All heuristics perform worse than before, because for 5562 test cases ($\approx$ 18,22%) none of the heuristics found the optimal solution. For the nature inspired algorithms, the difference between table 2 and 3 is larger than for the deterministic ones, because they found the best solution more often and therefore had only few suboptimal test cases which had to be considered in this analysis.

Table 3: Average degradation from the **optimal** solution (in %).

SA	MCP	GA	ETF	DLS	ACO
0,707	4,019	1,039	4,209	3,937	0,723

Figure 1 shows the percentage of test cases, to which the observed heuristics were able to find optimal solutions. Again SA and ACO perform much better than GA and the deterministic algorithms. Out of those, DLS found more optimal schedules than MCP and ETF. While the difference between the nature inspired and the deterministic algorithms is quite large, the last-mentioned behave nearly in the same manner and differ by only $\approx$2,8%.

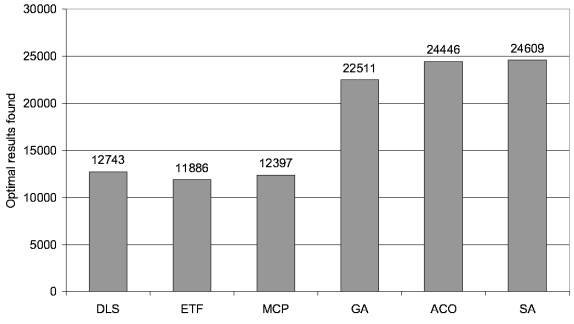

Fig. 1. Comparison considering the number of found optimal solutions.

As already mentioned above, the diverse structure of our test bench allows a thorough analysis of a heuristic's properties. The following results should give a first impression of possible examinations.

Figure 2 shows the effect of the target architecture's size on the quality of the heuristics' results. Obviously, SA and ACO perform better than all other heuristics. Except for target architectures with 2 processors, where SA finds $\approx$2,67% more optimal schedules. While ACO is almost unaffected by the target architecture's size, GA and the deterministic heuristics find optimal solutions more frequently, if the target architecture's size does not limit the parallelism of the schedule. Again, DLS performs better than MCP and ETF.

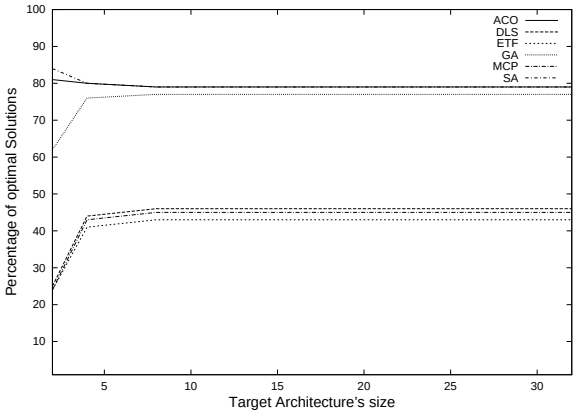

Fig. 2. Influence of the target architecture's size.

As can be seen in figure 2, SA is the only heuristic which finds more optimal schedules if the target architecture is small. Its success rate in finding an optimal solution drops from 84,21% for target systems with only two processors to approximately 80% for larger systems. Figure 3 shows, that for all target architecture sizes SA has clearly more difficulties in finding optimal schedules if the task graph has predominantly long edges. In contrast, DLS performs slightly better for task graphs with long edges if the target architecture is large.

Our investigation's next focus is the effect of the task graph's size on the quality of the found solutions. As can be seen in figure 4, the percentage of found optimal solutions decreases, when the task graphs' size increases. Nevertheless, SA and ACO scale better than GA which in turn performs much better than the deterministic algorithms. In contrast to SA, whose success rate drops from 87,42% to 72,58% (this is a difference of 14,84%), the MCP algorithm's success rate in finding optimal schedules drops by 52,36%.

With respect to the fact that most publications in literature use larger task graphs, with sometimes even more than 1000 tasks, the question arises, if the here presented

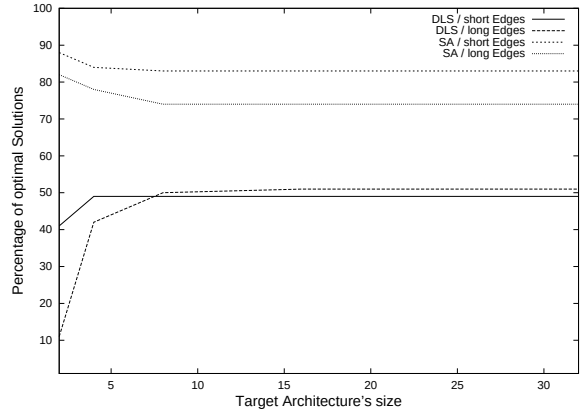

Fig. 3. Effect of the average edge length on SA's and DLS's success rate in finding optimal schedules.

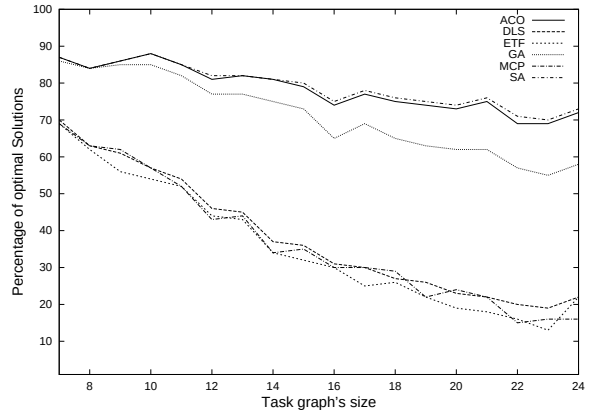

Fig. 4. Impact of the task graphs' size.

results are of any meaning to real world scheduling problems. In order to answer this question, we created another test bench which follows the same structure as the described one, but which consists of task graphs with up to 250 tasks. A representative test bench with larger task graphs would require too much hard disk space. First results with the large task graphs indicate, that the above observations will mainly hold for scheduling problems with larger task graphs (see table 4 for details).

4 Conclusion

In this paper, two different kinds of heuristic scheduling algorithms were compared: nature inspired stochastic and conventional deterministic algorithms. Our investigations are based on a comprehensive test bench that provides optimal schedules for 30511 test cases and therefore allows an elaborated comparison of heuristic algorithms. It could be clearly shown that the nature inspired algorithms outperform the conventional ones. Thus, future research in scheduling algorithms should pay more attention to this approach.

Table 4: Relative comparison with respect to larger task graph problems (up to 250 tasks).

	SA	MCP	GA	ETF	DLS
ACO	29056	34105	31904	34741	34194
	20303	9167	5570	6901	8462
	8753	24938	26334	27840	25732
DLS	34401	32645	34358	31746	
	29612	17320	19361	11505	
	4789	15325	14997	20241	
ETF	34750	33818	34652		
	30393	21120	22066		
	4357	12698	12586		
GA	32151	34294			
	29047	15687			
	3104	18607			
MCP	34354				
	28553				
	5801				

Acknowledgments: The authors would like to thank Mr. Markus Bank for developing the observed heuristics as part of his diploma thesis.

References

[1] Kwok, Y.-K., Ahmad, I. (1998) Benchmarking the Task Graph Scheduling Algorithms, Proceedings of the 12th International Parallel Processing Symposium, pp. 531–537

[2] Hönig, U., Schiffmann, W. (2004) A comprehensive Test Bench for the Evaluation of Scheduling Heuristics, Proceedings of the sixteenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), pp. 437–442

[3] Kwok, Y.-K., Ahmad, I. (1999) Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys, Vol. 31, No. 4, pp. 406–471

[4] Blum, C., Roli, A. (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys, Vol. 35, No. 3

[5] Dail, H., Casanova, H., Berman, F. (2002) A Decoupled Scheduling Approach for the GrADS Program Development Environment, Proceedings of the 2002 ACM/IEEE conference on Supercomputing, pp. 1–14