Discrepancy-Bounded Depth First Search

J. Christopher Beck and Laurent Perron

ILOG SA
9, rue de Verdun, BP85,
94253 Gentilly Cedex, France
{cbeck,lperron}@ilog.fr

Abstract. In this paper, we present a novel discrepancy-based search technique implemented as an instance of the generic search procedures framework introduced in [10]. Our empirical results indicate that the Discrepancy-Bounded Depth First Search (DBDFS) procedure exhibits a number of good properties. As a discrepancy based search technique, it is able to quickly find solution with low deviation from the optimal solution. In addition, it revisits fewer nodes than other discrepancy-based techniques (e.g., Limited Discrepancy Search), making it a good candidate for proving optimal solutions.

1 Introduction

Discrepancy-based search techniques such as Limited Discrepancy Search (LDS) [6] and Depth-bounded Discrepancy Search (DDS) [14] have been empirically shown to outperform traditional Depth First Search (DFS) in a number of problem contexts and theoretically in a simple model of search [6, 3, 14, 15]. The foundation for discrepancy-based search is the assumption that in any path from the root node to a solution node, a good heuristic is likely to make only a few mistakes. Therefore, paths with one "discrepancy" from the heuristic path should be searched before paths with two discrepancies, etc.

The evaluation of discrepancy-based techniques has tended to focus either on their ability to find satisfying solutions or on their ability to find solutions close to the optimal solution. When the goal of problem solving is to find and prove the optimal solution, DFS is often the favored search technique as it implements the optimal, worst-case behavior: DFS visits every node in the search space only once. In contrast, the usual implementations of discrepancy-based techniques incur an overhead due to the need to revisit internal nodes and, in some cases, leaf nodes [7, 14].

In this paper, we introduce and explore Discrepancy-Bounded Depth First Search (DBDFS), a novel discrepancy-based search technique motivated by the goal of finding and proving optimal solutions. DBDFS is designed to minimize the number of search states which are revisited while, at the same time, to preserve, as much as possible, the discrepancy-based order in which potential solutions are visited.

In the following section, we review generic search procedures and then describe DBDFS. Our empirical investigation, then, compares the performance of DBDFS with LDS, DDS, and DFS on two sets of job shop scheduling problems.

2 Search Procedures

A search procedure is a general framework for exploration of a search tree introduced in [10]. The search framework consists of the following components:

- A *node evaluator* which defines two methods:
  1. `evaluate`: a mapping, node $\rightarrow \mathbf{R}$. The result of the application of the `evaluate` method to a node is referred to as its *evaluation*.
  2. `subsume`: a comparison of the current node to the best current open node. If the method returns **true**, then the search engine postpones the evaluation of the two children of the current choice point and jumps to the best open node.
- A *priority queue* used to store the current set of open nodes, that is, the set of nodes that have been visited in the search but that have not had any of their children visited. A leaf node, as it has no children, cannot be an open node. The priority queue order is based on the ascending order of the evaluation of each node, with ties broken by choosing the node that was most recently visited during the search.

By specifying these two components, a wide variety of existing and novel tree exploration strategies can be easily experimented with.

3 Discrepancy-Bounded Depth First Search

Discrepancy-Bounded Depth First Search (DBDFS) performs depth first search on all nodes with paths from the root that have up to some bounded number of discrepancies. Given a bound of k, the ith iteration visits all leaf nodes with discrepancies between $(i-1)k$ and $ik-1$ inclusive. We refer to the bound, k, as the *width* of the search.

Figure 3, shows the order in which the leaf-nodes of a binary tree of depth 4 are visited using DBDFS with *width* = 2.

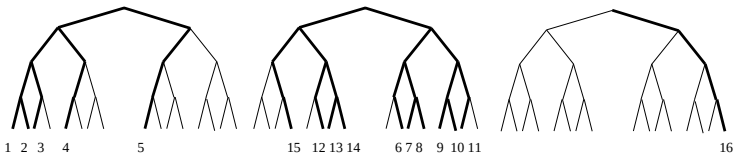

Fig. 1. DBDFS, $width$ = 2 on a Binary Tree of Depth 4 at iteration 1, 2 and 3

Using search procedures, we can define DBDFS simply using a node evaluator as shown in in algorithm 1.

4 Empirical Investigations

The goal of our empirical studies is to compare the performance of DBDFS with existing discrepancy-based techniques and DFS in problems where the goal is to find and

```
Procedure DBDFS::evaluate(Node node)
  d = node.discLevel
  if (d > maxDisc) then
    return ∞
  return (d / k)

Procedure DBDFS::subsume(Node node)
  return (bestEval < evaluate(node))
```

Algorithm 1: DBDFS as a node evaluator

prove the optimal solution. We use job shop scheduling as our domain of experimentation. Search procedures are implemented in ILOG Solver 4.4[13].

Optimization: A common form of optimization is to calculate an upper-bound on the optimization function, and then express this upper-bound as a problem requirement: no solution can have an evaluation greater than the upper-bound. If a solution is found with a lower evaluation, search is then restarted with a reduced upper-bound on the optimization function. In contrast, in this paper, we focus on *optimization by continuation*: at a solution state, rather than restarting search, the new bound is added at the leaf node. Search, then, continues as if the solution state was actually a failure. Prestwick [11] shows, with a number of artificial problem scenarios, that search by continuation for depth first search neither dominates nor is dominated by search by restart. However, he concludes that search by continuation is more robust as it avoids a number of pathological examples that result in exponentially more search effort for search by restart.

Scheduling Algorithms: All scheduling algorithms are implemented using ILOG Scheduler 4.4 [12]. As our main interest in these experiments is the comparison of DBDFS to existing search techniques, the different search techniques are the central independent variable in our experiments. The following search procedures are used:

- DBDFS2: Discrepancy-bounded Depth First Search with a width of 2.
- DBDFS4: Discrepancy-bounded Depth First Search with a width of 4.
- DFS: Depth First Search
- LDS: Limited Discrepancy Search
- DDS: Depth-bounded Discrepancy Search.

Note that only the two variations of DBDFS are implemented using the generic search procedure framework. In order to compare DBDFS against LDS and DDS, as they were originally defined, those techniques have been implemented according to the specifications in the original papers.

We hold all propagation techniques constant across all experiments (edge-finding (level 2 propagation), the precedence graph (level 1 propagation), and the sequence constraint [12]). We use two heuristic commitment techniques in different experimental conditions:

1. Texture: The texture-based heuristic is an implementation of the VarHeight heuristic commitment technique [3]. It sequences pairs of activities.

2. Rank: The rank heuristic picks the resource with minimum slack and completely ranks all activities on it before moving to the one with the next lowest slack [1, 4].

These two heuristics were chosen for their differences. The texture-based heuristic uses significant effort at each search state to identify a commitment while the rank heuristic is much less expensive. On finding satisfying solutions to job shop scheduling problems, the extra effort expended by the texture heuristic pays off in terms of being able to solve more problems in less CPU time [4].

Evaluation Criteria: We use two problem sets. The first is a set of 48 problem instances from the Operations Research library: ft06, ft10, ft20, la01-30, abz5-9, and orb1-10 [2]. The second set of problems consists of 50 job shop problems of size 15×15. These problems are machine-correlated, that is, the durations of the activities that execute on the same resource are drawn from the same distribution. Distributions differ for each resource. The following parameters are used for generation: σ_{lb} = 1, σ_{ub} = 10, μ_{lb} = 35, α = 0.75. See [15] for the meaning of these parameters and to access the generator.

For each job shop scheduling problem, we set the initial upper-bound on the makespan to be the sum of the durations of all activities. If a solution is found with a lower makespan, the new bound is added at the leaf node. Search, then, continues as if the solution state was actually a failure. Each algorithm is given a 20 minute CPU time limit within which to find and prove the optimal makespan. All experiments are run on a Sun UltraSparc10 with 256 MB of memory, running SunOS5.7.

5 Experimental Results

In Figures 2 and 3 we present two views of the results from the experiment using the OR library problems and the rank heuristic. In the first graph, we present the fraction of the experimental problems for which the optimal solution was found and proved within the limited CPU time of the experiment. While the two variations of DBDFS outperform the

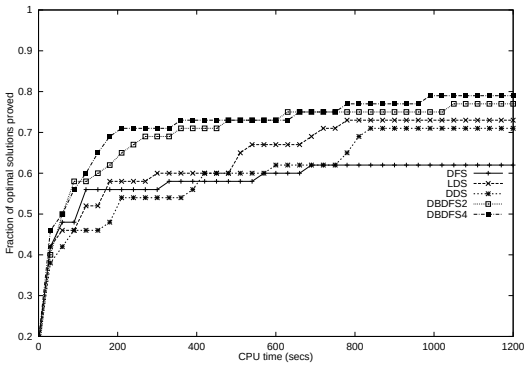

Fig. 2. Proved Solution for Rank Heuristic on OR Library Problems

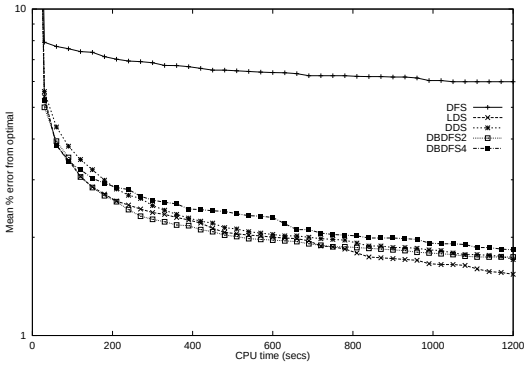

Fig. 3. Mean Deviation for Rank Heuristic on OR Library Problems

other search techniques, the quality of the other discrepancy-based techniques should also be noted. Despite the overhead in revisiting search states, both LDS and DDS are able to find and prove more optimal solutions than DFS. In Figure 3 we present the mean deviation from the optimal solution achieved by each search procedure (note the log-scale on the vertical axis). Again, we observe that the discrepancy-based techniques considerably outperform DFS. Among the discrepancy-based techniques, DBDFS2 appears slightly superior early in the search, but is overtaken by LDS as the search time increases.

The graphs in Figures 4 and 5 present the results generated by applying the texture heuristic to the OR library problems. In Figure 4, we see more uniform performance than in Figure 2: while DBDFS4 is slightly superior, there is not a large difference among the different search techniques. In particular, DFS performs much better than with the rank heuristic, even outperforming DDS. In Figure 5 the mean deviation from

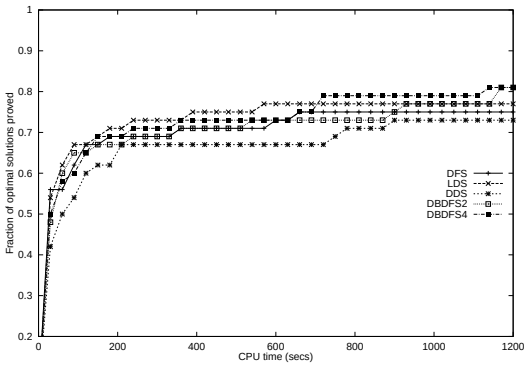

Fig. 4. Proved Solution for Texture Heuristics on OR Library Problems

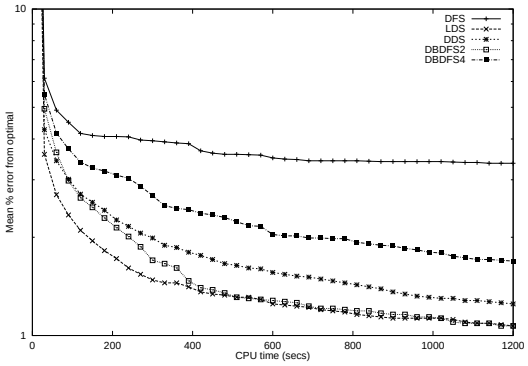

Fig. 5. Mean Deviation for Texture Heuristic on OR Library Problems

optimal again shows the superiority of the discrepancy-based techniques, with the best performance achieved by DBDFS2 and LDS.

Turning to the machine-correlated problems, Figures 6 and 7 display the results using the rank heuristic. Note that Figure 7 presents the percent deviation from the best known solution of each problem as the optimal solutions are not known. DFS performs very poorly in both graphs, failing to prove the optimal solution for any of the problems while achieving a mean deviation of just less than 15%. The other search techniques achieve much better, with a slight advantage for DBDFS2 and LDS in both graphs.

Finally, the results for the machine-correlated problems and the texture heuristic are presented in Figures 8 and 9. DFS is able to perform much better using the texture heuristic, however, it is still unable to match the performance of the discrepancy-based techniques. In terms of proving the optimal solution, DBDFS2 achieves the best performance, however it does not appear to be significantly better than the other discrepancy-

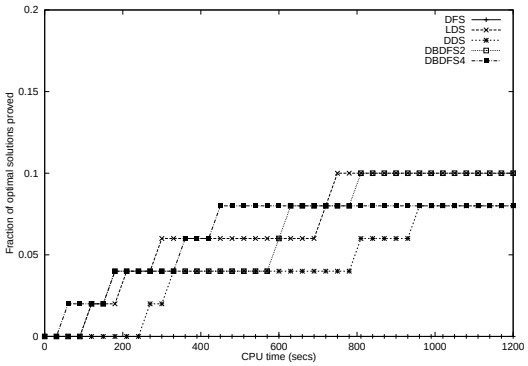

Fig. 6. Proved Solution for Rank Heuristic on machine-correlated problems

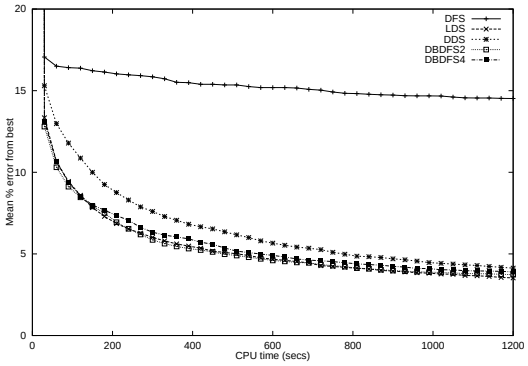

Fig. 7. Mean Deviation for Rank Heuristic on machine-correlated problems

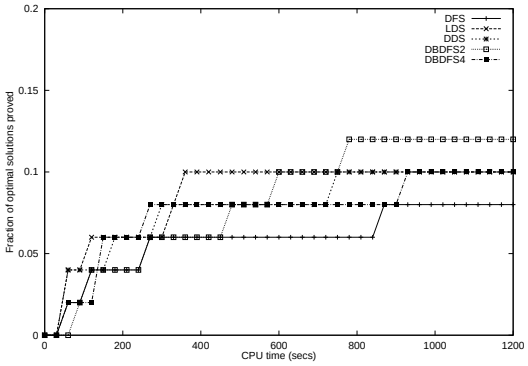

Fig. 8. Proved Solution for Texture Heuristics on machine-correlated problems

based techniques. For the mean percent deviation from best solution, we again observe a significant improvement in DFS using the texture heuristic. Overall, however, the discrepancy-based techniques are superior, with the best performance coming from LDS.

6 Discussion

Overall, it appears that DBDFS is competitive with existing discrepancy-based techniques both in terms of the ability to find and prove optimal solutions and in the ability to find good, suboptimal, solutions. Our original conjecture, however, was that we would be able to outperform existing discrepancy-based techniques by minimizing the revisiting of states while preserving the discrepancy-based search ordering.

The experiments in this paper do not support this conjecture. We attribute this lack of support to the surprisingly good performance of DDS and LDS in terms of proving

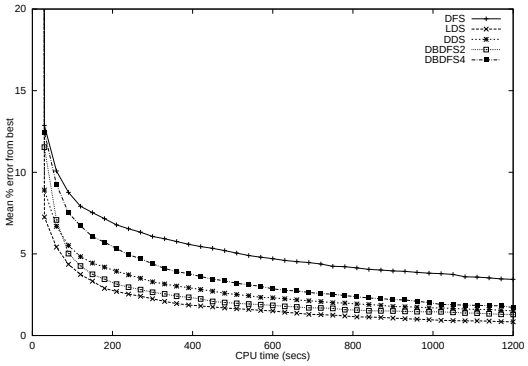

Fig. 9. Mean Deviation for Texture Heuristic on machine-correlated problems

optimal solutions. We believe that in domains which exhibit less back propagation than job shop scheduling, we will see superior performance of DBDFS. This is an interesting area for future work. An additional area for future work arises from the fact that DBDFS embodies a specific trade-off between discrepancy-based ordering and minimization of the revisiting of search nodes. While the DBDFS trade-off does not result in significant gains, a different balancing of these two factors may still achieve superior performance, even in the presence of strong back propagation. We intend to use the generic search procedure framework to perform a more formal investigation of this trade-off.

We divide the balance of our discussion into three parts: performance in finding and proving optimal solutions, performance in finding good solutions, and other comments.

Finding and Proving Optimal Solutions: The most interesting result from these experiments in terms of finding and proving the optimal solution is the quality of the discrepancy-based techniques. Despite the overhead that these techniques require in revisiting search states they are all able to outperform DFS in almost all experiments. We conjecture that this is due to the ability of discrepancy-based techniques to quickly find good solutions combined with the "back propagation" resulting from good solutions. Quickly found, good solutions allow much of the search tree to be pruned by the constraint propagation techniques. This pruning more than compensates for the overhead in revisiting states. Further investigation of this conjecture can be found in [5].

Turning to a comparison of the discrepancy techniques, we see that DBDFS is clearly superior only on the OR library problems with the rank heuristic (Figure 2). Good performance (comparable to LDS) is achieved in all other experiments. DDS, in contrast, appears to be the weakest of the discrepancy-based techniques in all graphs except Figure 8 where it appears to be superior of DBDFS4. Given previous work showing that DDS is often superior to LDS [14], we do not, as yet, have an explanation for these results. We conjecture that the existence of strong constraint propagation as well as back propagation from the optimal solution have a role to play in an explanation.

Finding Good Solutions: In terms of the ability to find good (but not necessarily optimal) solutions, previous work indicates that discrepancy-based techniques have a

strong advantage over DFS. Our experiments also show such an advantage as across all experiments as DFS finds solutions of significantly poorer quality. The poor performance of DFS is particularly evident with the rank heuristic, but is also observed when the texture heuristic is used.

In comparing discrepancy-based techniques, LDS appears to be able to consistently find solutions as good or better than the other techniques. Nonetheless, DBDFS2 is able to achieve performance that is quite close to LDS. Larger differences are seen with the texture heuristic, for example in Figure 5 where both DDS and DBDFS4 appear to be significantly worse than the other discrepancy-based techniques. Again we note that the poor performance of DDS in comparison with LDS is interesting but, as yet, unexplained.

Other Comments: The *width* parameter on DBDFS serves to control the trade-off between discrepancy-based ordering of search nodes and revisiting of search states. With a larger width, the search will more resemble DFS while with a smaller width, the discrepancy-based ordering is more heavily weighted. The effect of the width can be seen in comparing DBDFS4 with DBDFS2. In a number of experiments (e.g., Figures 2 and 4) DBDFS4 is able to find and prove more optimal solutions than DBDFS2. In contrast, DBDFS2 is better at finding good solutions.

The most obvious observation in comparing the performance of the search techniques with differing heuristics is that the texture heuristic improves DFS much more than the other search techniques. While all techniques achieve better performance with texture than with rank, the improvement in performance of DFS is disproportionate. Given the common belief that discrepancy-based techniques are better able to exploit good heuristics, this is an interesting observation. From another point of view, this observation indicates that the quality of the rank heuristic is improved much more than that of the texture heuristic when discrepancy-based techniques are used instead of DFS. This observation has been previously made in the context of CSP [3, 8].

A weakness of the experiments presented here is that they do not include Interleaved Depth First Search (IDFS) [9] in the performance comparison. IDFS, in fact, is particularly relevant to our conjecture as to the trade-off between discrepancy-based ordering and minimization of revisiting. IDFS exhibits significant non-locality when it "switches context" between search threads. Each context switch requires a jump in the search tree, revisiting many states in order to reestablish the context of the thread which is being jumped to. In our experience, such jumping degrades performance of the search technique. Therefore, the performance of IDFS represents a good test of our conjecture, and forms an important piece of future work.

7 Conclusion

We have introduced a novel discrepancy-based search technique, Discrepancy-Bounded Depth First Search (DBDFS), motivated by trying to minimize the overhead through the revisiting of states yet still maintain search through solutions by increasing number of discrepancies. Experiment comparing DBDFS to Limited Discrepancy Search, Depth-bounded Discrepancy Search and Depth-First Search indicate that all of the discrepancy-based techniques are much superior to DFS both in terms of finding and

proving optimal solutions and in terms of finding solutions with low mean deviation from optimal. While DBDFS is competitive and, in some cases, superior to LDS and DDS, we observed few significant differences in performance among discrepancy-based search techniques.

References

1. P. Baptiste, C. Le Pape, and W. Nuijten. Constraint-based optimization and approximation for job-shop scheduling. In *Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95*, 1995.
2. J. E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. Also available by ftp from ftp://graph.ms.ic.ac.uk/pub/paper.txt.
3. J. C. Beck. *Texture measurements as a basis for heuristic commitment techniques in constraint-directed scheduling*. PhD thesis, University of Toronto, 1999.
4. J. C. Beck, A. J. Davenport, E. M. Sitarski, and M. S. Fox. Texture-based heuristics for scheduling revisited. In *Proceedings of Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press, Menlo Park, California, 1997.
5. J. C. Beck and L. Perron. Proving optimality with complete stochastic and discrepancy-based search techniques. submitted to AAAI 00.
6. W. D. Harvey. *Nonsystematic backtracking search*. PhD thesis, Department of Computer Science, Stanford University, 1995.
7. R. Korf. Improved limited discrepancy search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.
8. C. Le Pape and P. Baptiste. An experimental comparison of constraint-based algorithms for the preemptive job shop scheduling problem. In *CP97 Workshop on Industrial Constraint-Directed Scheduling*, 1997.
9. P. Meseguer. Interleaved Depth-First Search. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 1382–1387, August 1997.
10. L. Perron. Search procedures and parallelism in constraint programming. In J. Jaffar, editor, *Proceedings of the Fifth International Conference on Principles and Practice of Constraint Programming (CP99)*, pages 346–360. Springer-Verlag, 1999.
11. S. Prestwich. Three CLP implementations of branch-and-bound optimization. In I. de Castro Dutra, M. Carro, V. Costa, G. Gupta, E. Pontelli, and F. Silva, editors, *Parallelism and Implementation of Logic and Constraint Logic Programming*. Nova Science Publishers, Inc, 1999. ISBN 1-56072-673-3.
12. Scheduler. *ILOG Scheduler 4.4 Users' Manual and Reference Manual*. ILOG, S.A., 1999.
13. Solver. *ILOG Solver 4.4 Users' Manual and Reference Manual*. ILOG, S.A., 1999.
14. T. Walsh. Depth-Bounded Discrepancy Search. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 1388–1393, August 1997.
15. J. Watson, L. Barbulescu, A. Howe, and L. Whitley. Algorithms performance and problem structure for flow-shop scheduling. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 688–695, 1999.