Local Search for Heuristic Guidance in Tree Search

Alexander Nareyek[1] and **Stephen F. Smith** and **Christian M. Ohler**[2]

Abstract. Recent work has shown the promise in using local-search "probes" as a basis for directing a backtracking-based refinement search. In this approach, the decision about the next refinement step is based on an interposed phase of sampling possible (but not necessarily feasible) variable assignments by local search. This information is then used to decide on which refinement to take, i.e., as a kind of variable- and value-ordering strategy.

In this paper, we investigate the efficiency of this hybrid search approach in the combinatorial domain of job-shop scheduling. First, we evaluate methods for improving probe-based guidance, by basing refinement decisions not only on the final assignment of the probe-construction phase but also on information gathered during the probe-construction process. We show that such techniques can result in a significant performance boost.

Second, we consider the relative strengths of probe-based search control and search control that is biased by more classically motivated variable- and value-ordering heuristics (incorporating domain-specific knowledge) that are not based on local search. Our results indicate that — while probe-based search performs better than an uninformed search — use of domain-specific knowledge is a much more effective basis for search control than information about constraint interactions that is gained by local-search probes, and leads to substantially better performance.

This paper provides only a brief overview. For a detailed presentation, please have a look at [2].

1 Introduction

A broad range of combinatorial problems is naturally formulated as constraint satisfaction problems (CSPs), and as such, the design of efficient and general search techniques for solving CSPs has attracted much attention in recent years. The major paradigms to solve CSPs are *refinement search* (or "tree search") and *local search*.

In refinement search, a stepwise reduction in the value domains of the decision variables is performed, interleaved with propagation phases, until each variable's domain has exactly one admissible value. If a refinement that was made turns out to be inconsistent later on, backtracking is applied to choose another refinement option. Search methods that are based on local search, on the other hand, generate a solution by repeatedly revising a concrete value assignment of the variables.

Refinement and local search have complementary strengths, and it seems useful to combine them. Use of local search can inject a global perspective to refinement search's control decisions; refinement search can provide a systematic basis for exploring tightly constrained regions of the underlying search space. In this paper, we focus specifically on use of the inconsistency measure of local search as guidance for determining which refinements to apply within a refinement search. First, we consider the added benefit of factoring information related to the local search's assignment history into search-control decisions. Second, we consider the relative strengths of probe-based heuristics in relation to uninformed refinement search and a search control bias that is provided by more classical heuristics that incorporate knowledge of the problem domain at hand. We investigate these issues in the application domain of job-shop scheduling.

2 The Solvers and Their Integration

The base refinement solver is built using the Comirem planning and scheduling framework [3]. The process of scheduling aims to feasibly sequence the set of tasks that are competing for each resource. A basic refinement step in the search involves two decisions: (1) selecting an as yet unsequenced task and (2) selecting where to insert this task into the partial sequence that has been established thus far on the required resource's timeline.

At each refinement step, the base refinement procedure computes the feasible insertion options for each uninserted task and applies search control heuristics to determine the next decision. When an infeasible state is detected, the search backtracks chronologically and considers alternative options for previously inserted tasks. Thus, the basic procedure is complete if given enough time, and search control heuristics are used to improve average case performance.

The local-search solver is a modification of the DragonBreath engine[3] (see [1] for details). The solver is a general constraint-programming system and not specialized to scheduling problems. However, it is easy to express the job-shop scheduling problem by way of the given constraint types.

In the search approach of the DragonBreath engine, every constraint calculates an inconsistency value using a constraint-specific measure indicating how far off the involved variables' assignment is from a consistent solution. The sum of the constraints' inconsistencies represents the total inconsistency of the current variable assignment. In each improvement iteration, an inconsistent constraint is selected, which then selects one of its constraint-specific heuristics to change the assignment in order to reduce the constraint's inconsistency. In contrast to the refinement solver, the local-search solver has concrete values assigned for all variables — the tasks' starting times in this case — at any time.

Figure 1 shows the interaction of the solvers. Refinement search is the master process, using the local search for heuristic guidance. Both solvers have different internal representations of the scheduling problem and communicate in reference to decision variables, which

[1] Cork Constraint Computation Centre, University College Cork, Cork, Ireland, e-mail: alex@ai-center.com

[2] School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891, USA, e-mail: sfs@cs.cmu.edu, ohler@cs.cmu.edu

[3] The engine is freely available at: http://www.ai-center.com/projects/dragonbreath/

are the tasks' starting times. For every refinement decision, the local-search solver is notified to internally add a corresponding constraint <4> so that both solvers keep working on the same problem. In case of backtracking, this constraint is removed again <5>.

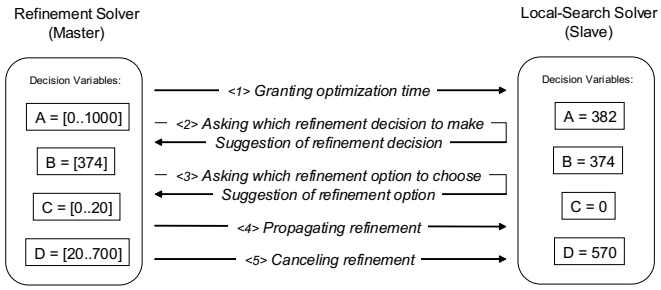

Figure 1. Solver integration.

The local-search solver can suggest, at each refinement step, which refinement decision to make and which refinement option to choose. For the job-shop scheduling problem, the local-search solver suggests for which task an ordering decision should be made next <2> and which ordering constraints (before or after) relative to other tasks on the same machine should be chosen <3>.

At each refinement step, the local-search solver is given some time for optimization <1> to adapt to the current situation (if constraints were added or removed) before a recommendation is queried.

3 Different Strategies for Local-Search Advice

The following strategies to provide advice for guiding the refinement solver's search decision are tested:

- **Strategy `CI`: Current Inconsistency**
  This strategy only exploits the information that is available after the improvement phase of local search has been completed. After a local-search phase, it is evaluated if the potential refinement decisions would cause inconsistencies. A refinement decision is recommended that steers away from the option that causes the highest inconsistency.
- **Strategy `II`: Iterations of Inconsistency**
  During a local-search phase, this strategy keeps track of the number of iterations in which potential refinement decisions cause inconsistencies. A refinement decision is recommended that steers away from the option that causes an inconsistency most often.
- **Strategy `AI`: Average Inconsistency**
  Not only the number of iterations with inconsistencies is considered, but also the average of the inconsistency for these iterations. The recommendation is based on this average instead of the only the number of iterations with inconsistencies.

In addition, the refinement-search solver can propagate changes to the domain bounds of individual start-time variables to the local-search solver. Bounds represent hard constraints in the DragonBreath engine, i.e., a variable's value will immediately be shifted back into the bound if a heuristic tries to assign a value outside the bound. **Strategy** `AI-B` is `AI` with bound propagation.

For every problem instance and solving strategy, 50 test runs are computed. Strategies are then compared in a pairwise way for each problem, rating a strategy as better if it was able to find a solution (within the cut-off limit) in 10% more cases than the other strategy.

Figure 2 shows the results for all strategy pairs. It can be seen that `II` outperforms `CI`, and `AI` performs best of all; this confirms that the exploitation of inconsistency information gathered during the local-search phase can substantially improve local search's guidance.

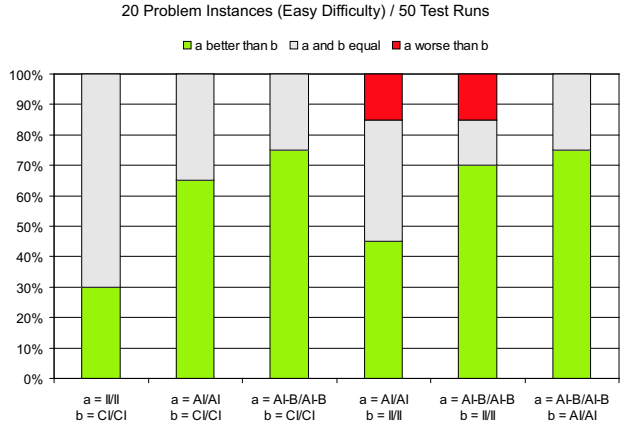

Figure 2. Comparing variants of local-search recommendations.

Propagating bounds — strategy `AI-B` — is nearly always an improvement over strategy `AI`. However, keep in mind that we count refinement steps/nodes and not computation time. Propagating bounds to the local-search solver adds computational overhead, and whether this pays off depends on the solvers and communication speed.

4 Comparison to Other Forms of Advice/Heuristics

In comparison to simple uninformed/randomized choices for task selection and ordering, the local-search advice performs much better (see main paper [2]). However, the costs of running a local-search solver must be considered. In our experimental setting, it turned out to be a substantial overhead that hardly justifies the improvement.

For many problems, it may be easy to come up with domain-specific refinement heuristics. Our results indicate that even simple heuristics of this kind substantially outperform the local-search advice (see [2]). This seems to be plausible, but we were surprised how easy it is to outperform the probe-based heuristic. These results are all the more significant considering that use of the refinement heuristics is *much* cheaper than calling the local-search solver.

For other domains than job-shop scheduling, where less domain knowledge can be exploited by refinement heuristics, local-search-based guidance may still be a viable option. In these cases, one should consider the extended integration techniques presented in this paper, i.e., also incorporating information gathered during the local search phase and feeding propagation results back into local search.

Acknowledgements: This work was supported in part by DARPA Contract #F30602-00-2-0503 and the CMU Robotics Institute.

REFERENCES

[1] Nareyek, A. Using Global Constraints for Local Search. In Freuder, E. C., and Wallace, R. J. (eds.), *Constraint Programming and Large Scale Discrete Optimization*, American Mathematical Society Publications, DIMACS Volume 57, 9–28, 2001.

[2] Nareyek, A.; Smith, S. F.; Ohler, C. M. Integration of a Refinement Solver and a Local-Search Solver. CMU Tech Report, TR–CMU–RI–04–33, Pittsburgh, PA, USA, 2004.

[3] Smith, S.F.; Hildum, D.; and Crimm, D. Interactive Resource Management in the Comirem Planner. *Proceedings AAAI Workshop on Mixed-Initiative Intelligent Systems*, Acapulco Mexico, August 2003.