Infeasible Search Analysis for Oversubscribed Scheduling Problems

Mark F. Rogers
Computer Science Dept., Colorado State University
Ft. Collins, CO 80523 USA
rogersma@cs.colostate.edu

Introduction

Researchers have expended considerable effort developing local search neighborhoods and heuristics that restrict search to feasible space. However, some have discovered that a variety of problems may be solved efficiently if an algorithm includes both feasible and infeasible solutions in its search. In a procedure called *strategic oscillations* (Glover 1989), Glover defined a local search strategy that alternates a search between feasible and infeasible states and uses a *span* to control the number of moves the search makes in either region. An algorithm may increase or decrease its span periodically to vary the search intensity in feasible and infeasible regions.

Strategic oscillation algorithms appear to hold two advantages over feasible-only search: first, the oscillations tend to focus a search around the feasible-infeasible frontier, or *boundary region*, where optimal solutions reside for many constrained optimization problems (Glover 1989; Schoenauer & Michalewicz 1996). Second, some researchers claim that infeasible search opens new routes to optima and permits an algorithm to exploit short-cuts in the space (Glover 1989; LeRiche, Knopf-Lenoir, & Haftka 1995; Kelly, Golden, & Assad 1993; Michalewicz 1996).

The boundary region is defined as feasible states that have infeasible neighbors, or infeasible states that have feasible neighbors. When a search reaches a boundary-region state, its next move may introduce or resolve constraint violations as the search moves into the opposite region. Thus boundary-region search is motivated by the conjecture that optima for constrained optimization problems frequently include *binding* or *active* constraints. A constraint becomes active whenever a variable reaches its minimum or maximum allowed value. Research has verified that boundary region search finds good solutions for domains such as laminate design (LeRiche & Haftka 1994), numerical optimization (Schoenauer & Michalewicz 1996) and single-machine scheduling (Hurink & Keuchel 2001).

That a search may benefit from short-cuts is an appealing concept, but until now we have seen no direct established evidence that infeasible search finds paths that could not be reached as directly using feasible solutions. In addition, the concept of short-cuts appears to conflict with the goal of boundary-region search; to uncover efficient paths through infeasible space, a search must move away from the boundary region, effectively postponing boundary exploration until the search returns from infeasible space. We have confirmed that a strategic oscillation algorithm finds short-cuts through infeasible space, but it is difficult to justify extensive infeasible search.

We have experimented with two satellite scheduling domains that each consist of a single *oversubscribed* satellite: each problem instance has many more requests than the satellite can accommodate. The first domain is the 2003 ROADEF Challenge problems: satellite scheduling problems designed for a competition (Cung 2003). We have applied a strategic oscillation algorithm, $Tabu_{CL}$ (Cordeau & Laporte 2003) to the ROADEF problems and to a synthetic earth observing satellite (EOS) domain. In both domains the best solutions are likely to contain tasks with active constraints. The search spaces are large: ROADEF has $n! \cdot 2^n$ possible states in a problem with n tasks; for EOS the number of states is bounded by $n! \cdot m^n$, where n is the number of tasks and m is the maximum number of time windows allocated to a task. Both domains include time window constraints which we relax in order to introduce infeasible solutions into a search.

Infeasible Path Segments

We define a *path segment* as any sequence of schedules explored during a search. A *feasible* segment is one in which each schedule is feasible. An *infeasible* segment is one where each schedule in the segment contains at least one constraint violation except for the start and end schedules, which are feasible. This restriction on the start and end schedules helps us divide infeasible search into distinct infeasible segments.

We further distinguish between infeasible segments that stay within the boundary region and those that move outside the boundary region. We define a *boundary-region* segment as an infeasible segment whose states all reside along the boundary. When at least one infeasible state has no feasible neighbors, we refer to its segment as a *deep* infeasible segment. These distinctions allow us to evaluate the relative merits of large and small oscillation span values, and provide insights into how likely infeasible search is to yield significant benefits.

To assess the efficiency gained through infeasible search, we consider three phenomena: cycles, detours and short-

cuts. When a segment's start and finish state are identical, then the segment is a *cycle*. If there is a cycle contained within an infeasible segment from one infeasible state to another, then we identify the segment as a *detour*. Cycles and detours degrade search performance by wasting moves. The final infeasible segment we consider is the *short-cut*: an infeasible segment whose length is shorter than any feasible segment between two feasible states.

We have conducted experiments to test the hypothesis that infeasible solutions facilitate efficient search by opening short-cuts in a search space. For our oversubscribed scheduling problems, we have found that short-cuts do exist for segments that leave the boundary region, but their efficiency may be offset by infeasible segments that yield little or no benefit to a search.

Tabu Search

The $Tabu_{CL}$ implementation uses three neighborhood operators: `insert` that inserts an image into the schedule; `remove` that removes an image from the schedule, and `replace` that changes the image that will be acquired. The `insert` operator finds the first possible location for an image by examining each possible slot in the schedule. Thus a schedule is biased towards placing tasks in the earliest possible slot. This strategy ignores possible time window violations for images other than those immediately adjacent to the candidate image, so an `insert` operation may generate infeasible schedules.

The `replace` operator attempts to remove an image from the schedule and to replace it with an equivalent image from the same request. Thus the `replace` operator may also introduce time window violations whenever the transition times or the time window for an alternate image differ from those of the original.

$Tabu_{CL}$ strategic oscillations with a simple linear penalty function and a scale factor, α. For a schedule s, a profit value $p(s)$, and time window violation count $w(s)$, the evaluation function $f(s)$ is given by:

$$f(s) = p(s) - \alpha w(s) \quad (1)$$

As a search progresses, the value α controls oscillations about the boundary region. As a search moves through feasible space, α decreases stochastically, thus encouraging the search to admit infeasible schedules. Once the search enters infeasible space, α begins to increase and eventually forces the search back toward feasible schedules.

We applied a modified version of this algorithm to the EOS problems using the same neighborhood operators. The EOS problems use a different objective function than ROADEF, but the strategic oscillation behavior is the same.

To account for the boundary region's influence, we implemented a restricted version of the tabu search ($Tabu_{BR}$) that prevents a search from moving away from the boundary. We compared $Tabu_{CL}$ with $Tabu_{BR}$, to see how much a search benefits from boundary-region search and how much it gains from traversals away from the boundary.

Infeasible Segment Analysis

During each run, whenever a search transitioned from a feasible state to an infeasible state, we recorded the feasible state and each move in the infeasible path segment. When the search transitioned again back to feasible space, we recorded the ending feasible state and allowed the search to continue until it started another infeasible segment.

We conducted infeasible path analysis using a separate program to assess each infeasible segment recorded during a search. To categorize infeasible segments, we implemented an algorithm that detects the characteristic infeasible segment types defined in the previous section. In addition to categorizing infeasible paths we also differentiate between boundary region traversals and deep traversals that move beyond the boundary region.

Results

We have confirmed the existence of short-cuts for algorithms that include infeasible states. Our infeasible path statistics confirm that short-cuts, cycles and detours all occur during infeasible search. However, the value of deep infeasible trajectories varies between problems.

When we use strategic oscillations to allow a search to probe deeply into infeasible regions, we find that in most cases, deep traversals generate a higher proportion of improving moves than boundary-region search. At the same time, they tend to enter fewer cycles, and they uncover a higher proportion of short-cuts in the space. Each of these attributes implies that deep infeasible traversals have the potential to make a search highly efficient.

However, when we compare $Tabu_{BR}$ with $Tabu_{CL}$ on the ROADEF problems we do not find consistent performance improvements. Possibly this is because short-cuts still comprise a minority of deep traversals (roughly 20-40%), while deep traversals themselves consume more iterations than boundary-region traversals. Thus while deep traversals may occasionally yield promising short-cuts, this benefit is offset by the number of iterations squandered on non-improving deep path segments. To address this issue, we have begun exploring ways to eliminate unproductive segments.

Infeasible Tabu List

Our first approach was to augment $Tabu_{CL}$ with a tabu list for infeasible paths. With this enhancement we have been able to eliminate most of the cycles and detours found in infeasible traversals. We hoped that an infeasible tabu list would also force a search to explore a more diverse selection of infeasible paths than the original search and thus exploit more shortcuts. Although the enhanced algorithm nearly doubled the proportion of shortcuts for the ROADEF problems, we saw little improvement for most problems and the new tabu list tended to degrade performance overall.

For the EOS problems we found that an infeasible tabu list made the search more efficient than the original search in some cases, but made little difference in others. Evidently it is not sufficient merely to increase the proportion of short-

cuts: in addition we need to consider where a shortcut begins in a space, and whether it finds an improving path.

Future Work

Infeasible path metrics should give us new ways to assess strategic oscillation search. By creating tools that enable us to measure infeasible search attributes, we obtain more comprehensive algorithm performance metrics than we would by merely charting an objective function during a run. By examining the relationships between algorithm features and infeasible search metrics, we should be able to improve these algorithms further or demonstrate that improvements are unlikely. Ultimately our goal is to identify a "best" algorithm for a given problem domain, but there are still questions we want to answer regarding the ROADEF and EOS domains.

Although an infeasible tabu list yielded good results for some problems, it provides little insight into search behavior that leads to unimproving shortcuts. We want to examine the states where we find cycles, detours and unproductive shortcuts to see if there are trends common to these phenomena that will allow us to eliminate a proportion of wasted moves. As our results show, the difference between boundary region search and deep infeasible search can be subtle; eliminating even a small number of detrimental moves could improve search performance.

"Jump" Tabu

Our $Tabu_{CL}$ ROADEF results revealed that for many infeasible paths longer than two moves, the best moves in each neighborhood changed little as the search proceeded into infeasible space. We are investigating the possibility we could make $Tabu_{CL}$ more efficient simply by eliminating the neighborhood searches for some initial infeasible steps. By sorting the best moves in the initial infeasible step and then applying several of them at once, we could "jump" ahead in the search by generating the same infeasible path using fewer evaluations.

A naïve approach is simply to apply k of these moves at the start of each infeasible path to make a jump. The average infeasible path length for ROADEF problems ranges from 2 to 5 moves and of these, approximately half will be `insert` or `replace` moves at the start of an infeasible segment. Thus the simplest implementation applies the first $k = 2$ moves from an initial infeasible neighborhood, skipping one set of evaluations. A more sophisticated approach will attempt to predict an optimal number of moves to make in a single jump as a function of the penalty parameter value, the current gain and the violation counts for the best infeasible moves.

The "jump" approaches may allow us to reduce wasted evaluations by applying multiple moves at once. We also wish to determine whether it is possible to predict when an infeasible traversal is likely to produce a productive shortcut. At this point, our infeasible path statistics do not reveal any obvious trends. Currently we are investigating whether a machine learning tool such as a C4.5 classifier could identify subtle indications that lead to productive infeasible paths. If we can construct such a classifier, then we may gain further insight into what causes unproductive infeasible paths, or incorporate the classifier directly into a search.

Neighborhoods and Algorithms

To date, our work has focused on $Tabu_{CL}$ and its performance with two oversubscribed scheduling problems. Our results hint that an algorithm's neighborhood operators may dictate shortcuts' frequency and influence during local search. Thus we want to expand our investigation to examine shortcut behavior with different neighborhood operators and algorithms.

In $Tabu_{CL}$, neighborhood operators are not symmetric: it may not be possible to undo a move in a single step. For example, if we `remove` a task and immediately `insert` the same task, the resulting schedule may change. The algorithm greedily schedules tasks at the earliest possible time, so a task that is removed and reinserted may move to an earlier slot in the schedule. We hypothesize that this kind of asymmetry may explain the large number of shortcuts we find with $Tabu_{CL}$. Thus we would like to know how relevant symmetry is for finding shortcuts whether algorithms that use symmetric neighborhoods can find shortcuts as well.

If we are able to identify search characteristics that lead to productive infeasible search, a related goal will be to apply our results to other search algorithms such as simulated annealing or genetic algorithms. Research has shown that strategic oscillation strategies are compatible with temperature schedules in simulated annealing (Anagnostopoulos *et al.* 2003). In addition, researchers have applied strategic oscillations successfully to genetic algorithms in a number of domains (for example, (Joines & Houck 1994; Coit, Smith, & Tate 1996; Eiben & Ruttkay 1996; Bean & Hadj-Alouane 1997)).

Strategic oscillations can be an effective method for solving constrained optimization problems by relaxing problem constraints to introduce infeasible solutions into a search. Infeasible space provides access to both sides of the boundary region and allows a search to focus its attention on good solutions that may reside there. However if we want to develop efficient search algorithms, then we need to understand what characteristics make infeasible search successful. By studying infeasible path statistics, we hope to uncover the infeasible search features that yield insights into how a search behaves and may thus allow us to exact the benefits while avoiding the unproductive forays.

References

Anagnostopoulos, A.; Michel, L.; Hentenryck, P. V.; and Vergados, Y. 2003. A simulated annealing approach to the traveling tournament problem. In *Proceedings of CPAIOR 2003*.

Bean, J. C., and Hadj-Alouane, A. B. 1997. A genetic algorithm for the multiple-choice integer program. *Operations Research* 45(1):92–101.

Coit, D. W.; Smith, A. E.; and Tate, D. M. 1996. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing* 8(2):173–182.

Cordeau, J. F., and Laporte, G. 2003. Maximizing the value of an earth observation satellite orbit. Technical Report CRT-2003-27, Centre de recherche sur les transports.

Cung, V.-D. 2003. ROADEF'2003: Results of the final stage (base X) of the challenge. `http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2003/results030203_final.html`.

Eiben, A. E., and Ruttkay, Z. 1996. Self-adaptivity for constraint satisfaction: Learning penalty functions. In *International Conference on Evolutionary Computation*, 258–261.

Glover, F. 1989. Tabu search–Part I. *ORSA Journal on Computing* 1(3):190–206.

Hurink, J. L., and Keuchel, J. 2001. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics* 112(1-3):179–197.

Joines, J. A., and Houck, C. R. 1994. On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GA's. In *International Conference on Evolutionary Computation*, 579–584.

Kelly, J. P.; Golden, B. L.; and Assad, A. A. 1993. Large-scale controlled rounding using tabu search with strategic oscillation. *Annals of Operations Research* 41:69–84.

LeRiche, R., and Haftka, R. T. 1994. Improved genetic algorithm for minimum thickness composite laminate design. In *Proceedings of the International Conference on Composite Engineering, Aug 28–31*.

LeRiche, R.; Knopf-Lenoir, C.; and Haftka, R. T. 1995. A segregated genetic algorithm for constrained structural optimization. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 558–565. Morgan Kaufmann Publishers Inc.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. London: Springer.

Schoenauer, M., and Michalewicz, Z. 1996. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature IV*, 245–254. Berlin: Springer.