

Looking for Shortcuts: Infeasible Search Analysis for Oversubscribed Scheduling Problems

Mark F. Rogers and Adele Howe and Darrell Whitley

Computer Science Dept., Colorado State University

Ft. Collins, CO 80523 USA

{rogersma,howe,whitley}@cs.colostate.edu

Abstract

Searches that include both feasible and infeasible solutions have proved to be efficient algorithms for solving some scheduling problems. Researchers conjecture that these algorithms yield two primary benefits: 1) they tend to focus on solutions close to the boundary between feasible and infeasible solutions, where active constraints are likely to yield optimal values, and 2) moves that include infeasible solutions may uncover short-cuts in a search space. Researchers have published empirical studies that confirm the value of searching along the feasible-infeasible boundary, but until now there has been little direct evidence that infeasible search yields short-cuts.

We present empirical results in two oversubscribed scheduling domains for which boundary region search in infeasible space appears to offer advantages over search in strictly feasible space. Our results confirm that infeasible search finds shortcuts that may improve search efficiency more than boundary region search alone. However, our results also reveal that inefficient infeasible paths which we call *detours* may degrade search performance, potentially offsetting efficiency shortcuts may provide.

Introduction

Researchers have expended considerable effort developing 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 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 assess whether a strategic oscillation algorithm really does find short-cuts through infeasible space and if so, whether the short-cuts will yield enough to justify more extensive infeasible search.

For our investigation we consider 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). Cordeau and Laporte implemented a strategic oscillation tabu search algorithm (*Tabu_{CL}*) that performed exceptionally well (Cordeau & Laporte 2003). We use their algorithm as the basis for our work. We apply this algorithm to the ROADEF problems and to problems from 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 (Cordeau & Laporte 2003); 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.

Within infeasible segments, we further distinguish between those 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 state in an infeasible segment has no feasible neighbors, we refer to the 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 (see Figure 1). When a segment's start and finish state are identical, then the segment is a *cycle*. Although *Tabu_{CL}* maintains a tabu list, the tabu mechanism is only designed to prevent cycles for feasible segments. In addition to simple cycles, if there is a cycle 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 in this study is the *short-cut*. We define a short-cut as an infeasible segment whose length is shorter than any feasible segment between two feasible states.

We conducted experiments to test the hypothesis that infeasible solutions facilitate efficient search by opening short-cuts in a search space. To our knowledge, this is the first study to provide empirical evidence that short-cuts occur in infeasible search. In addition, we wanted to determine whether strategic oscillations are efficient when they are allowed to travel away from the boundary region into infeasible space, or if they are more efficient when they remain close to the boundary region. For our oversubscribed scheduling problems, we 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.

Background

For strategic oscillation search in a tabu search framework, Glover has observed that "moving outside of a boundary and returning from different directions uncovers opportunities for improvement that are not readily attainable when

the search is more narrowly confined" (Glover 1989). He suggests that a tabu list forces a search to alternate between improving and non-improving moves and in some cases, between feasible and infeasible solutions.

Critical event tabu search (Glover & Kochenberger 1996) extends strategic oscillation algorithms by focusing on *critical events*, when a search is in the boundary region and no improving moves can be found among the feasible neighbors. Upon reaching a critical event, the search progresses to a solution in infeasible space and checks any feasible neighbors for new optima before continuing into infeasible space. A counter tracks the number of infeasible moves, and when the counter reaches the *span*, the search is forced back toward feasible space until it encounters another critical event. By thus intensifying a search around the boundary region, critical event tabu search has been shown to be effective for constraint optimization problems such as multiconstrained and multidimensional knapsack (Glover & Kochenberger 1996; Oppen, Gruner, & Løkketangen 2003).

A common method for controlling oscillations is to incorporate a penalty into the objective function: infeasible solutions yield objective values that may be better than feasible solutions, and the penalty permits comparison between infeasible and feasible solutions. In some cases the penalty increases monotonically during a search to yield a series of diminishing infeasible segments. In other cases the penalty increases and decreases to induce span oscillations.

Kelly et al. (1993) describe a tabu search implementation that uses strategic oscillations to solve controlled rounding problems. They adjust the span as search progresses: for a given span, a scalar penalty increases and decreases with a period of eight values per oscillation. The authors state that strategic oscillations succeed in part because "reaching the optimal solution may require a long tortuous path through feasible space, whereas if a solution path is allowed to enter infeasible regions, then the optimum can be found rather easily."

LeRiche et al. (1995) explored genetic algorithms that use infeasible solutions to find composite laminate designs. Rather than establishing a static penalty value, they tuned penalty functions for their domain.

Kathryn Dowsland (Dowsland 1998) describes an elaborate local search strategy she used to construct rosters for a nursing staff at a large hospital. Oscillations take place in three phases: phase 1 consists of a complex series of moves through infeasible space that search for a feasible solution; phase 2 applies a subset of the methods from phase 1 in an attempt to improve the feasible solution, and phase 3 escapes local optima by allowing the search to return to infeasible space. The author reports good results for the strategic oscillation search and compares her results to simpler, canonical straw-man algorithms: random-descent, simulated annealing and tabu search. Although canonical search algorithms perform respectably on synthetic problems, their performance degrades significantly on real data, while the strategic oscillation implementation consistently finds optimal or near-optimal solutions.

Amaral and Wright explored the effectiveness of a strategic oscillation tabu search algorithm on manufacturers'

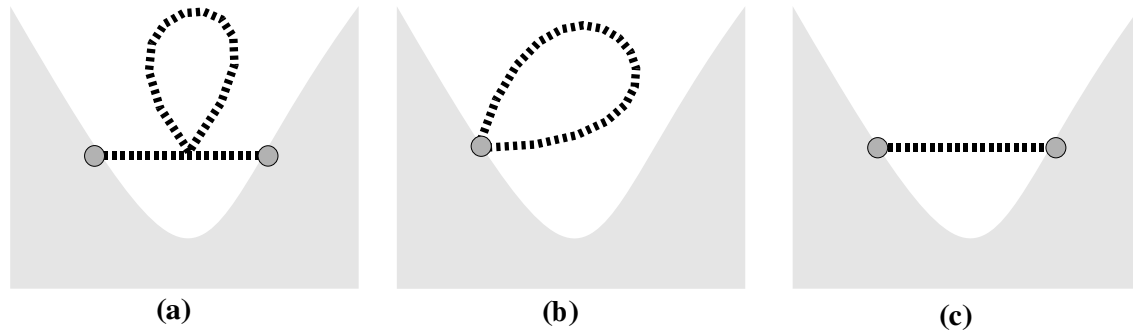


Figure 1: Three infeasible segment types: (a) a *detour* encounters the same state twice before returning to feasible space, (b) a *cycle* begins and ends at the same state, and (c) a short-cut cannot be duplicated in the same number of feasible moves.

pallet-loading problems (MPLP). Their algorithm consists of two phases similar to critical-event tabu search: there is a constructive phase that admits infeasible solutions, and a destructive phase that eliminates constraint violations. They report performance that makes their algorithm an attractive alternative to an efficient algorithm that produces exact solutions (Amaral & Wright 2001).

Cordeau and Laporte implemented a strategic oscillation tabu search algorithm that placed high in the 2003 ROADEF competition (Cung 2003; Cordeau & Laporte 2003). The authors had applied a similar algorithm to vehicle routing problems (Cordeau, Larporte, & Mercier 2001) so its success in the ROADEF competition demonstrated the algorithm’s robustness in multiple domains. The authors implemented a penalty parameter that increased stochastically while the search explored infeasible space and decreased when the search entered feasible space.

Barbulescu et al. (2004) present results for an oversubscribed scheduling application in which the final proposed schedules are infeasible but provide good initial solutions for human schedulers to refine. Their algorithm explores infeasible schedules away from the boundary region: most solutions are multiple moves away from feasible space.

Kramer and Smith report good solution quality and execution speed with repair heuristics for oversubscribed scheduling problems. These algorithms systematically relax and tighten priority constraints, “temporarily creating ‘infeasible’ solutions in hopes of arriving at a better feasible solution.” (Kramer & Smith 2003; 2005) Thus their build-and-repair strategies behave like strategic oscillation methods, though they focus on boundary region efficiency.

Each of the algorithms that permits infeasible traversals showed competitive advantages over alternatives. While the authors implied or stated that infeasible search contributed to the performance, none analyzed whether it was due strictly to boundary region exploration or whether the algorithms were finding short-cuts.

Most research that seeks to explain infeasible search efficiency has focused on boundary-region search. A few researchers mention the possibility that short-cuts will yield search efficiency. Those who do mention short-cuts (such

as (LeRiche, Knopf-Lenoir, & Haftka 1995; Smith & Tate 1993; Kelly, Golden, & Assad 1993)) do not provide empirical evidence to show whether they occur. Finding short-cuts likely depends on how far a search is allowed to move into infeasible space and on the search space characteristics specific to a given domain.

Satellite Scheduling Domains

In striving to understand why algorithms such as $Tabu_{CL}$ are so efficient, we begin by duplicating $Tabu_{CL}$ for the ROADEF domain to investigate short-cuts and then apply the same method to a similar domain to see if our ROADEF results generalize. Thus both domains in this study are single-resource satellite scheduling problems. In these domains we seek to schedule multiple image acquisition requests on a single earth observing satellite. Each problem has many more requests than can be scheduled. Each request has an associated priority or profit. The goal is to maximize the value of the requests we can accommodate in a schedule. In both domains, we relax time window constraints to introduce infeasible solutions into a search.

Every request provides two or more opportunities to acquire an image, thus increasing problem complexity. As we schedule requests in both domains, we must account for transition times between adjacent images that introduce gaps in the schedule. Transition times may be longer than image acquisition times and may dominate a schedule, so good schedules tend to minimize transition times.

ROADEF

The 2003 ROADEF Challenge introduced a set of oversubscribed satellite scheduling problems with a single resource (satellite) that processes image acquisition requests. There are two kinds of requests: a *target* that consists of a single image, and a *polygon* that consists of two or more images. An image’s profit value is proportional to the square surface area it represents. A target image’s profit is fixed for a given problem and directly reflects the target’s size. A polygon’s profit is a piecewise-linear function of the proportion of its surface area that is included in the schedule.

To acquire images, the satellite carries a camera and rotates it between image captures. The camera may acquire an image by scanning either forward or backward. In addition to the time required to capture each image, there is a transition time that accounts for repositioning the camera between two acquisitions. Depending on how tasks are scheduled, the transition between two tasks can take more time than the actual image acquisitions. Thus an optimal schedule not only includes as many requests as possible, but also minimizes the transition times between images.

Images may be either mono or stereo. Stereo images are represented as two separate requests for the same target (or polygon), with an added constraint that either both requests must be in the schedule, or neither request is in the schedule. When a stereo request is included in a schedule, both images must be acquired in the same direction.

The ROADEF Challenge data set consists of three sets of 10 problem instances: an “A” set of test instances; a “B” set that was introduced for the competition’s qualifying round, and an “X” set used in the final competition (Cung 2003). The “A” data set consists of instances that range from trivial problems that contain fewer than 10 images to large problems containing over 1,000 images. The “B” and “X” sets each consist of problem instances containing 600 to 1,000 images. Problem complexity changes for each problem, as the relative proportion of targets to polygons and stereo to mono images changes.

EOS

EOS single-satellite problems are similar to the ROADEF problems in many respects. The goal is to schedule image capture requests on a satellite. The satellite circles the earth in a sun-synchronous orbit at an altitude of 800 kilometers. The satellite carries an imaging sensor and a solid-state recorder (SSR) capable of storing up to 75 images at a time. The sensor faces straight down, but is capable of rotating 24 degrees to either side of its track. The time required to rotate the sensor is called *slew* time. The slew rate, measured in degrees per second, determines how influential transition times will be for a given problem. Image acquisitions take 24 seconds, so the slew rate has a profound effect on how much transition times influence final schedules.

The SSR has a fixed capacity and stores images on board until the satellite passes over a remote tracking station, when it can dump its data and reset its capacity. Once a satellite’s SSR reaches capacity, the satellite cannot accommodate image acquisition requests until it has dumped its current data.

Globus et al. (2004) generated problems that included 2100 targets for a satellite to ensure that the problems were oversubscribed. Unfortunately, *Tabu_{CL}*’s best-first approach took a prohibitive amount of time to perform an iteration with 2100-target problems. Thus although we used similar criteria to construct realistic problems for our experiments, we reduced the simulated problems by a factor of 10 to make our experiments manageable. For each problem, we randomly selected 210 land-based targets from the Geographic Nameserver Database (NGIA 2004) and used STK (AGI 2004) to model 16 hours’ worth of orbits and target

visibilities. We created a set of 10 problems and applied *Tabu_{CL}* to find solutions.

There are several key differences between the ROADEF and the EOS problems. The ROADEF problems add complexity with polygon images and include stereo images that impose additional constraints. The EOS problems impose constraints on the number of images that may be acquired between SSR dumps. We use two different evaluation functions for the two domains: ROADEF uses a piecewise-linear function of each task’s profit, while EOS measures task priorities and image quality. Despite these differences, both problems provide an opportunity to introduce infeasible solutions by admitting time window violations. Thus if the *Tabu_{CL}* algorithm was successful in the ROADEF domain, it is likely to be effective for EOS as well.

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 direction in which an image will be acquired. The *insert* operator uses a greedy strategy that 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 the same image acquired from the opposite direction. Thus the *replace* operator may also introduce time window violations whenever the transition times or the time window for a reversed image differ from those of the original. Whenever a *remove* or *replace* operation removes an image from a schedule, *Tabu_{CL}* records the image in a tabu list to prevent cycling.

The algorithm applies 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:

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

When a schedule is feasible, it has no time window violations, so $w(s) = 0$ and $f(s) = p(s)$. The profit value $p(s)$ is the unpenalized evaluation function for this domain. For every request in a schedule, $p(s)$ assigns a profit value proportional to the request’s images that appear in the schedule. The proportional profit values are computed using a piecewise linear function that penalizes small proportions and rewards large proportions.

As a search progresses, the value α controls oscillations about the boundary region. Whenever a search moves to a feasible solution, α is reduced using $\alpha = \alpha/\delta$; when the search moves to an infeasible solution, α is increased using $\alpha = \alpha\delta$, where $\delta = 1 + U(0, 1)$ is a uniform random value between 1 and 2. As a search progresses through feasible

space, α decreases, 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 apply a modified *Tabu_{CL}* algorithm to the EOS problems using the same *insert*, *remove* and *replace* neighborhood operators. The EOS problems use a different objective function than ROADEF; the function assesses image quality as well as the value of scheduled images. To incorporate these characteristics into a single function, we use the objective described in (Globus *et al.* 2004): the objective is to minimize the total value of unscheduled images, minimize the average image distortion and minimize the amount of time spent moving the image sensor. An image’s value is determined by its priority and by the image quality. Image distortion is estimated using the imaging sensor slew angle: high angles are associated with severe distortion. In addition, this implementation incorporates a penalty term to permit the search to explore infeasible solutions.

To evaluate a schedule s , we compute the total value of unscheduled tasks, $u \in U$, plus the average slew time and the average slew angle. When these factors are taken into account, the multi-objective evaluation function becomes:

$$f(s) = w_p \sum_{u \in U} P_u + w_s \mu_{slew} + w_a \mu_{angle} + \alpha w(s) \quad (2)$$

μ_{slew} and μ_{angle} are the average slew time and image angle, and $\sum_{u \in U} P_u$ is the sum of the unscheduled request priorities. Consistent with Globus *et al.*, we use weights $w_p, w_s, w_a \in [0, 1]$ to dictate the relative importance of image priority, slew time and image angle, respectively. For our experiments, we set $w_p = 1, w_s = 0.01$ and $w_a = 0.02$. As in the ROADEF *Tabu_{CL}* implementation, α is a dynamic, stochastic penalty parameter that adjusts the relative weight of the penalty function $w(s)$, a simple time window violation count.

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 region. By comparing *Tabu_{CL}* with *Tabu_{BR}*, we can see to what extent a search benefits from boundary-region search and how much it gains from traversals away from the boundary.

Tracking Infeasible Paths

To understand how infeasible search works for *Tabu_{CL}*, we ran the algorithm on each of the ROADEF and EOS problem instances. Every time 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 read and assess each infeasible path segment recorded during a search. To categorize infeasible segments during the assessment phase, we implemented an algorithm

that detects the characteristic infeasible segment types defined in the previous section: short-cuts, cycles and detours. Cycles are easy to detect since the start and end schedules are identical.

Recall that we define a detour as an infeasible segment that contains a cycle but does not have the same start and end state. Thus to detect detours, we simply extended the cycle detection strategy to look at all pairs of states along an infeasible segment. If any pair of states are identical, then the segment between them constitutes a cycle in infeasible space, and we identify the segment as a detour.

If we identify a segment as a cycle, it cannot possibly be a short-cut, but if we identify a detour, it is possible that the segment is still shorter than any feasible path between the same start and end states. Thus for any segment that is not a simple cycle, we apply our short-cut detection algorithm.

We base the short-cut detection method on the following observation: if we assume that an infeasible segment is a short-cut, then its moves represent a shorter segment between two solutions than would be possible in feasible space. Further, these moves uniquely identify the tasks that must be manipulated to achieve the solution at the end of the segment; a feasible path will have to apply the same moves to achieve the same end schedule, using a different order to avoid infeasible states.

For every infeasible segment, our algorithm tries every permutation of moves in the set in an attempt to construct a similar segment through feasible space. If at any point during this process a feasible segment arrives at the desired end state, then the infeasible segment cannot be a short-cut. If no series of feasible moves reaches the end state, that implies that at least one additional move would be required to do so: the infeasible segment thus represents a short-cut.

There are $n!$ possible permutations for a path segment consisting of n moves, so currently the algorithm tries permutations only for segments having no more than eleven moves (up to 4×10^7 permutations per segment). In general this is not a severe restriction; while collecting infeasible path statistics for the ROADEF problems we found that infeasible segments average between two and five moves, depending on the problem; overall, 98% have fewer than eleven moves. For all but two of the problems, at least 60% of deep infeasible segments are shorter than eleven moves.

Distinguishing BR and Deep Paths

In addition to identifying short-cuts we also differentiate between boundary region traversals and deep traversals that move beyond the boundary region. For an infeasible traversal to move away from the boundary region, it must include states that have no feasible neighbors. During search we flag any state that has no feasible neighbors and record its segment as a deep segment.

Results

We wish to answer several questions about infeasible search behavior. How frequently do we get cycles, detours and short-cuts in infeasible search? Do these frequencies change between boundary-region path segments and deep segments? Does infeasible search find good solutions? Does

$Tabu_{CL}$ benefit from exploring deep infeasible paths or does $Tabu_{BR}$ perform equally well?

Search Spaces

Both satellite scheduling domains have search spaces that are dominated by infeasible states. The reason for this is that since these problems are oversubscribed, at any point during a search there will be a multiple tasks competing for the resource at different times in the schedule.

We found that if we attempt to insert tasks randomly into a schedule, inserted tasks are likely to push scheduled tasks outside their feasible time windows within a few insertions. Using a Monte Carlo approach we found that except for the simplest problems, fewer than 2% of randomly-constructed ROADEF schedules were feasible, and 3% for EOS. Almost every feasible schedule is on the boundary region (has infeasible neighbors): for any scheduled task, it is nearly always possible to find a set of unscheduled tasks that compete for the same time window. We were also able to confirm that good solutions have active constraints. For the ROADEF problems, on average 30% of the images in a good schedule had at least one active constraint; for EOS, the proportion jumped to over 90%.

In both domains, $Tabu_{CL}$ reaches an equal number of feasible and infeasible solutions. When the α parameter has increased enough to force the search back into feasible space, it remains high for several moves, forcing the search to continue moving to feasible solutions.

Does Infeasible Search Find Short-Cuts?

As the results in Table 1 show, we were able to confirm the existence of short-cuts for all of the ROADEF problems. We were surprised at the high proportion of short-cuts found for some problems: for example, in ROADEF problem instance 2-28-155, nearly half of all infeasible path segments were short-cuts. The EOS results (Table 2) confirm the existence of short-cuts in that domain as well; in fact, the proportion is much higher than for ROADEF.

For the ROADEF domain, we can see that infeasible search yields a relatively small number of improving segments, which may be due to a high proportion of cycles. Cycles comprise a large proportion of the segments, which is not surprising, since the $Tabu_{CL}$ tabu mechanism only tracks images that have been recently removed from a schedule. Infeasible segments tend to have image removals only at the end, so they are unlikely to benefit from the tabu implementation. A typical infeasible cycle will insert a few images and immediately remove the same images as it is forced to return to feasible space. Although we expected similar behavior with detours, we found that detours comprise only a small proportion of infeasible segments.

The statistics for the EOS problems show that there is a much lower percentage of cycles for these problems and a higher proportion of improving segments and short-cuts than we found in the ROADEF domain. We also notice that the average infeasible segment length is longer.

For deep path segments we see the same trends as in ROADEF, though the differences are more subtle. The proportion of improving segments remains consistent while the

proportion of cycles decreases and the proportion of detours rises. Again there is an increase in the proportion of short-cuts, but the change is slight.

Is There Efficiency Beyond the Boundary?

Table 2 reveals some striking differences for deep traversals. In general, the proportion of short-cuts increases; in some cases, the change is dramatic. For example, in problem 2-13-111, over 56% of the deep path segments are short-cuts, compared with 13% overall for that problem. The proportion of cycles drops dramatically for all problems, although the proportion of detours increases slightly. The number of improving moves increases for most problems, although 2-28-111 shows a noticeable drop. These results suggest that deep infeasible traversals have the potential to increase search efficiency despite the presence of cycles.

When we compared $Tabu_{BR}$ with $Tabu_{CL}$, we found few performance differences between the two for most of the ROADEF problems. However, Figure 2 shows examples of two problems that exhibit some improvement from one search to the other. For problem 2-28-66, $Tabu_{CL}$ was able to reach better solutions than $Tabu_{BR}$ although the boundary region search appeared to converge more quickly. For problem 3-28-155, $Tabu_{BR}$ consistently found better solutions than $Tabu_{CL}$.

For the EOS problems we found that $Tabu_{CL}$ performed slightly better than $Tabu_{BR}$ on seven problems while $Tabu_{BR}$ found the best solutions only for problem 10 (see Figure 3). Thus even when it appears that deep infeasible search may improve overall search efficiency for some problems, the search must strike a balance between allowing time to find good short-cuts and wasting evaluations on fruitless deep traversals.

Infeasible Tabu List

Experimental results show that infeasible paths tend to yield a high proportion of cycles. For the ROADEF problem, cycles comprise 65% of attempted infeasible paths, and for EOS, cycles represent 24%. Thus a simple way to reduce the number of cycles for $Tabu_{CL}$ in both domains is to add a second tabu list for infeasible paths. We hypothesized that by using a simple tabu list, we could eliminate most of the cycles and detours found in infeasible traversals. In addition, an infeasible tabu list could force a search to explore a more diverse selection of infeasible paths than the original search and thus exploit more shortcuts.

Infeasible traversals tend to consist of a sequence of `insert` operations that increase the number of violations, followed by `replace` or `remove` operations that reduce the number of violations. Cycles in infeasible paths thus tend to occur when the search inserts a set of images and immediately removes those same images. Thus our infeasible tabu list contains images inserted during an infeasible traversal and prevents the algorithm from removing those images as the search seeks a path back to feasible space.

Our infeasible tabu list is simpler than the original tabu list: it uses the same tenure values, but it does not include long-term memory or aspiration. As a further simplification,

Infeasible Path Statistics for ROADEF Problems										
	# Images	All Infeasible Paths				Deep Infeasible Paths				
		Avg. Length	Cycle	Detour	Shortcut	Deep	Avg. Length	Cycle	Detour	Shortcut
Mean	376	3.2	65%	2%	24%	10%	9.4	11%	6%	24%
Min	28	2.3	34%	0%	8%	2%	5.3	0%	2%	8%
Max	534	4.9	91%	3%	44%	25%	12.5	59%	10%	50%

Infeasible Path Statistics for EOS Problems										
	# Images	All Infeasible Paths				Deep Infeasible Paths				
		Avg. Length	Cycle	Detour	Shortcut	Deep	Avg. Length	Cycle	Detour	Shortcut
Mean	210	5.9	24%	4%	72%	29%	9.7	11%	7%	82%
Min	210	5.5	14%	2%	65%	24%	8.7	5%	3%	77%
Max	210	6.8	32%	5%	83%	34%	10.8	20%	11%	91%

Table 1: Infeasible path statistics for the ROADEF and EOS problems. The first column shows the number of images in each problem. The second column shows the average infeasible segment length. The remaining columns show the proportions of cycles, detours and shortcuts among infeasible paths. The second half shows the proportion of deep infeasible segments, along with the proportions of each path type among the deep segments.

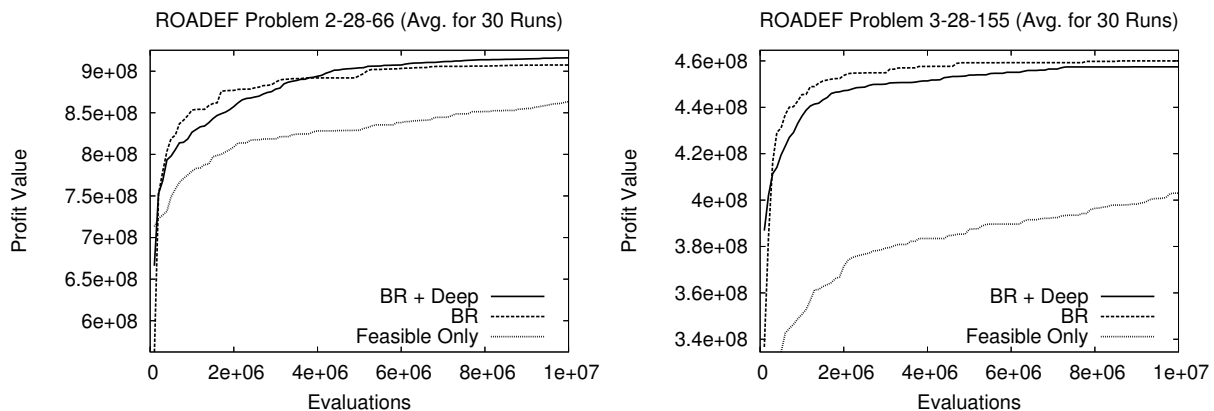


Figure 2: Two examples of ROADEF problems that exhibit different behavior with boundary region and deep infeasible search. For problem 2-28-66, a combination of BR and deep infeasible search yields greater efficiency than BR search alone. For problem 3-28-155, deep infeasible traversals do not add efficiency and appear to degrade performance.

the search begins each infeasible traversal with an empty infeasible tabu list; it does not retain tabu moves from one infeasible path to the next. To see how the infeasible tabu list impacted search performance, we repeated our experiments with the ROADEF and EOS problems.

Figure 4 shows how the infeasible tabu list performed on two different ROADEF problems. Although this algorithm was more efficient for problems such as 2-13-111, for most ROADEF problems (such as 2-15-170) the infeasible tabu list degraded performance. The EOS results shown in Figure 5 exhibit a striking difference from the ROADEF results: in some cases, such as problem 10, the infeasible tabu list tends to make the search much more efficient than the original search. Infeasible tabu search exhibited its worst performance on Problem 6, yet even in this worst case the new tabu list performed as well as the original search.

Table 2 shows the statistical results from the second set of runs. As we predicted, the proportion of cycles declined dramatically in both domains while the proportion of detours declined slightly. The infeasible tabu list nearly doubled the

proportion of shortcuts for the ROADEF problems, yet we saw little improvement in the results. Evidently it is not sufficient merely to increase the proportion of shortcuts: in addition we need to consider where a shortcut begins in a space, and whether it finds an improving path.

Discussion

With this study we have confirmed the existence of shortcuts 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 po-

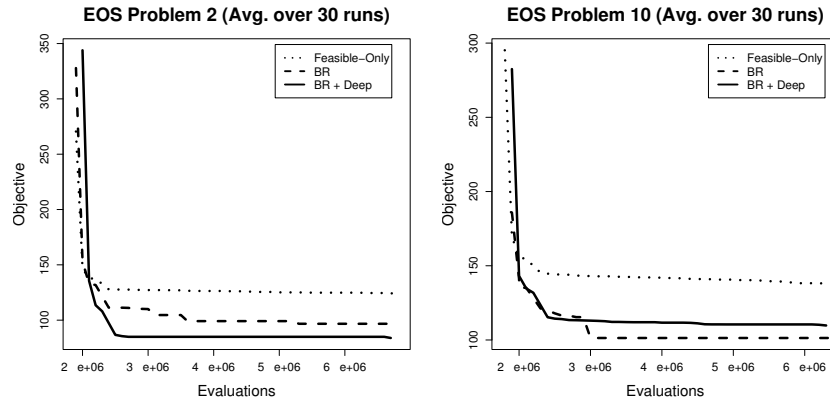


Figure 3: Two examples of our EOS problem results. For most problems, such as #2 above, a combination of BR and deep infeasible search yields greater efficiency than BR search alone. For problems such as #10, however, deep infeasible traversals appear to degrade performance.

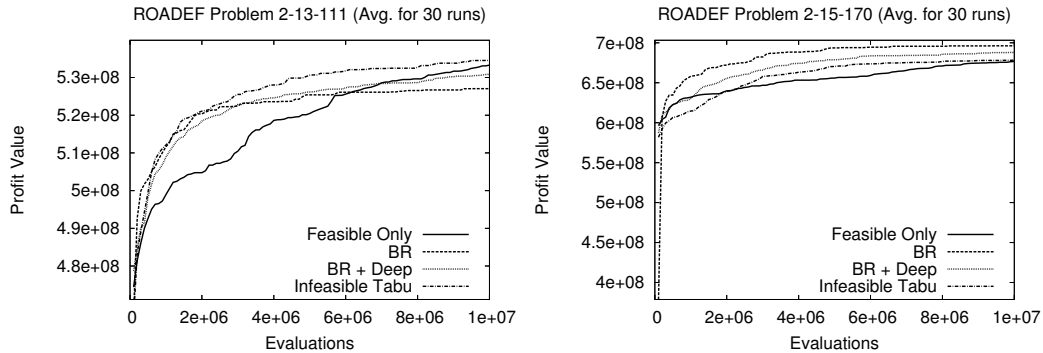


Figure 4: Examples of two ROADEF problems that exhibit different performance with an infeasible tabu list. The infeasible tabu list dominates other implementations for problem 2-13-111, but under-performs for 2-15-170.

tential to make a search highly efficient.

However, when we compare $Tabu_{BR}$ with $Tabu_{CL}$ on the ROADEF problems we do not find significant 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.

In contrast to the ROADEF problems, we did not find a significant difference between boundary-region traversals and deep traversals for the EOS domain. This may be due to the number of time windows in the two domains: for the ROADEF problems, every image has two time windows that typically overlap. The difference between the two time windows in the ROADEF domain is subtle, so once a search reaches good solutions, infeasible search may provide a way to allow small time window violations while the search looks for a better way to shuffle a subset of tasks.

In the EOS domain, a task’s time windows are completely separate and do not compete with one another, so the fine

granularity that we see in ROADEF problems may not occur in EOS problems. EOS may thus represent a class of problems for which boundary-region search yields more efficiency than deep infeasible search, while ROADEF is an example of a domain where occasional deep traversals can yield significant benefits.

When we added an infeasible tabu list, we mitigated cycles and detours in infeasible paths and increased the proportion of shortcuts found during a search. These enhancements yielded good results for the EOS problems, but performance varied on the ROADEF problems. Thus although we have confirmed that infeasible search can find shortcuts, shortcuts do not always improve search efficiency.

Future Work

We found that infeasible path metrics are useful tools for assessing strategic oscillation search. 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.

By creating tools that enable us to measure infeasible

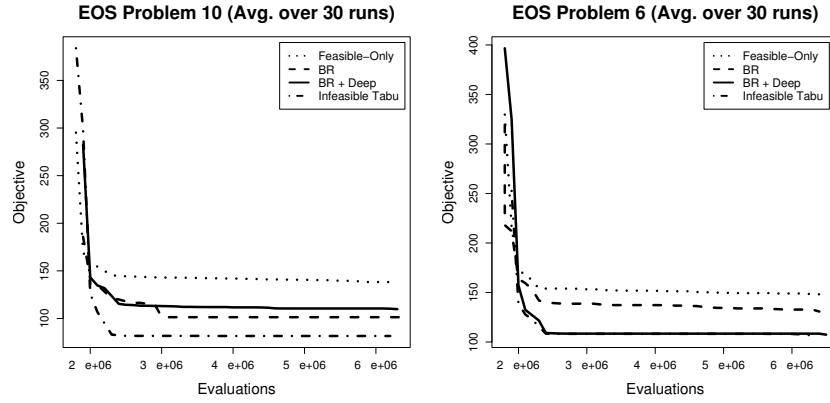


Figure 5: Infeasible tabu list performance for EOS problems. In cases such as #10, search with an infeasible tabu list dominates other methods. At worst (#6), the infeasible tabu list does not appear to degrade performance.

ROADEF Statistics with Infeasible Tabu List										
	# Images	All Infeasible Paths				Deep Infeasible Paths				
		Avg. Length	Cycle	Detour	Shortcut	Deep	Avg. Length	Cycle	Detour	Shortcut
Average	376	6.3	6%	2%	47%	10%	12.6	0%	1%	55%
Min	28	3.4	0%	0%	22%	3%	5.8	0%	0%	21%
Max	534	10.5	30%	8%	74%	18%	26.8	1%	6%	87%

EOS Statistics with Infeasible Tabu List										
	# Images	All Infeasible Paths				Deep Infeasible Paths				
		Avg. Length	Cycle	Detour	Shortcut	Deep	Avg. Length	Cycle	Detour	Shortcut
Average	210	7.2	1%	1%	72%	39%	10.2	0%	2%	84%
Min	210	6.7	1%	0%	65%	32%	9.3	0%	0%	74%
Max	210	8.2	2%	3%	75%	49%	11.2	1%	4%	92%

Table 2: Results for $Tabu_{CL}$ with an infeasible tabu list, averaged over all ROADEF and EOS problems.

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.

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.

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.

Acknowledgments

This research was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-03-1-0233. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Adele Howe was also supported by the National Science Foundation under Grant No. IIS-0138690. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors gratefully acknowledge Jean-François Cordeau of HEC Montréal for his guidance during our $Tabu_{CL}$ implementation. We also wish to acknowledge Mark Roberts of Colorado State University and Al Globus at NASA for their assistance with the EOS problem domain. Finally, we thank the anonymous reviewers for their helpful insights and comments on this work.

References

- Amaral, A. R. S., and Wright, M. 2001. Experiments with a strategic oscillation algorithm for the pallet loading problem. In *International Journal of Production Research*, volume 39, 2341–2351.
- Barbulescu, L.; Whitley, L. D.; and Howe, A. E. 2004. Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *Proceedings of the 19th National Conference on Artificial Intelligence*, 143–148.
- 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.
- Cordeau, J. F.; Laporte, G.; and Mercier, A. 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52:928–936.
- 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.
- Dowland, K. A. 1998. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operations Research* 106:393–407.
- Globus, A.; Crawford, J.; Lohn, J. D.; and Pryor, A. 2004. A comparison of techniques for scheduling earth observing satellites. In *Proceedings of the 19th National Conference on Artificial Intelligence*, 836–843.
- Glover, F., and Kochenberger, G. A. 1996. Critical event tabu search for multidimensional knapsack problems. In *Meta Heuristics: Theory and Applications*, 407–427. Kluwer Academic Publishers.
- 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.
- 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.
- Kramer, L., and Smith, S. 2003. Maximizing flexibility: A retraction heuristic for oversubscribed scheduling problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.
- Kramer, L., and Smith, S. 2005. Maximizing availability: A commitment heuristic for oversubscribed scheduling problems. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*.
- 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.
- Oppen, J.; Gruner, S.; and Løkketangen, A. 2003. A tabu search based heuristic for the 0/1 multiconstrained knapsack problem. In *Norwegian Informatics Conference Proceedings*.
- Schoenauer, M., and Michalewicz, Z. 1996. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature IV*, 245–254. Berlin: Springer.
- Smith, A. E., and Tate, D. M. 1993. Genetic optimization using a penalty function. In Forrest, S., ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufman.