RANDOM WALKS AND NEIGHBORHOOD BIAS IN OVERSUBSCRIBED SCHEDULING

Mark Roberts, L. Darrell Whitley, Adele E. Howe, Laura Barbulescu
Colorado State University

Abstract This paper presents new results showing that a very simple stochastic hill climbing algorithm is as good or better than more complex metaheuristic methods for solving an oversubscribed scheduling problem: scheduling communication contacts on the Air Force Satellite Control Network (AFSCN). The empirical results also suggest that the best neighborhood construction choices produce a search that is largely a greedy random walk of the graph induced by the complete neighborhood.

Keywords: Local Search, Heuristic Search, Real World Scheduling

1. INTRODUCTION

Local search and metaheuristic search methods that are used in conjunction with local search, such as Tabu search and simulated annealing, have proven to be both robust and effective across a wide range of combinatorial optimization problems. Yet our understanding of local search is largely intuitive: modifications to good solutions are likely to lead to the discovery of other good solutions. No Free Lunch proofs show that in the general case this intuition is wrong; local search is no better than random search over all possible functions (Wolpert and Macready, 1995). Of course, we generally assume (without proof) that the problems where local search is found to be effective have some inherent structure that is being leveraged by local search.

Another common assumption is that carefully crafted neighborhoods are generally better than more random neighborhoods. The choice of neighborhood is always a compromise. The neighborhoods need enough connectivity to assure that the space between random starting points and optima can be traversed in a reasonable number of steps. Over all possible functions, the expected number of local optima is determined by neighborhood size. For a search space of S points and a neighborhood of size k, the expected number of local optima for a randomly chosen function is $S/k+1$ (Whitley et al., 1997). Large neighborhoods result in few local optima in expectation. However, too large a neighborhood may mean too much time spent deciding on each step. Given n tasks to schedule, an $\mathcal{O}(n^2)$ neighborhood can be very costly to evaluate if one is using steepest descent local search.

This paper examines the effect of neighborhood choice on the performance of local search on a large real world application: scheduling the Air Force Satellite Control Network (AFSCN) (Barbulescu et al., 2004b). Approximately 500 contacts with earth orbiting satellites from a set of 16 antennas are scheduled in each 24 hour period. This problem has been shown to be $\mathcal{NP}$-complete (Barbulescu et al., 2004b). For the AFSCN scheduling problem with 500 tasks, the shift neighborhood is of size $\frac{500(499)}{2} = 124,750$. Yet, other methods, such as genetic algorithms and squeaky-wheel optimization (Joslin and Clements, 1999), find good solutions using less than 50,000 evaluations. Steepest descent local search with this neighborhood configuration is simply not competitive.

This paper examines the bias found in four variations on the shift neighborhood under next descent local search. We consider combinations of two binary characteristics (see Table 1): size and order. For

Size \ Order	Structured	Unstructured
Full	**N1**: Ordered	**N2**: Random Unrestricted
Restricted	**N3**: Interaction Restricted	**N4**: Random Restricted

Table 1. The neighborhood types discussed in this paper.

size, neighborhoods are either complete (the full $\mathcal{O}(n^2)$ neighborhood) or are restricted. For order, the neighborhoods are either structured in some way or they are random samples of larger neighborhoods. The restricted and structured neighborhood (N3) is designed to exploit task interactions that are known to impact schedule quality.

The results on 12 days of actual data show that local search with the right choice of neighborhood is just as effective as a genetic algorithm or squeaky-wheel optimization. What is more surprising is that the normal intuitions about what is the right choice of neighborhood do not hold for AFSCN scheduling. Randomly constructed neighborhoods drawn from the unrestricted full neighborhood yield the best performance. Furthermore, the best local search method appears to be nothing more than a greedy random walk of the graph induced by the search neighborhood.

2. AFSCN SCHEDULING

AFSCN scheduling consists of scheduling communication requests for earth orbiting satellites from a set of 16 antennas at 9 ground-based tracking stations. Customers submit requests that are scheduled by humans in a complex arbitration process. This problem is an instance of a single machine problem with release and due dates where the objective is to minimize the number of late jobs, denoted $1|r_j|\sum U_j$ in the machine scheduling literature (Pinedo, 2002). A formal specification and proof for $\mathcal{NP}$-completeness of this problem are found in (Barbulescu et al., 2004b). We provide here a quick introduction of the relevant problem characteristics.

Although AFSCN starts as an oversubscribed scheduling problem, all jobs are eventually scheduled through negotiating relaxed task requirements; so our automated scheduler reduces the effort of the human schedulers by minimizing one of two objective functions: 1) the total number of tasks in conflict or 2) the sum of overlaps between conflicting tasks.

In the current formulation, satellites are grouped according to their orbits: low-altitude and high-altitude. Figure 1 depicts an exemplar request for each altitude. Low-altitude requests (top) typically have short visibility windows (15 minutes) during which a single contact request can be scheduled; these tasks usually have few scheduling alternatives. In contrast, high-altitude requests (bottom) have longer durations (20 minutes or more) with much larger visibility windows. The scheduling alternatives can include different ground tracking stations. Both request types include information about which ground stations and times are possible alternatives.

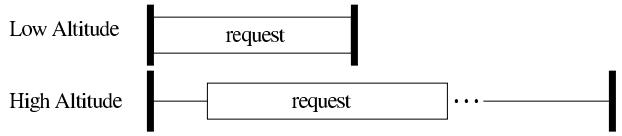

Figure 1. An idealized example of low-altitude (top) and high-altitude (bottom) requests in AFSCN. Low-altitude requests have short visibility windows (15 minutes) with few alternative resources. High-altitude requests have much larger visibility windows and are longer (20 minutes or more); these requests often have many alternative resources.

ID	Date	Size	# Low	# High	Best Conflicts	Best Overlaps
Day1	10/12/92	322	153	169	8	104
Day2	10/13/92	302	137	165	4	13
Day3	10/14/92	311	146	165	3	28
Day4	10/15/92	318	142	176	2	9
Day5	10/16/92	305	142	163	4	30
Day6	10/17/92	299	144	155	6	45
Day7	10/18/92	297	142	155	6	46
Mar07	03/07/02	483	225	258	42	773
Mar20	03/20/02	457	194	263	29	486
Mar26	03/26/03	426	183	243	17	250
Apr02	04/02/03	431	185	246	28	725
May02	05/02/03	419	178	241	12	146

Table 2. Problem characteristics for the 12 days of AFSCN data used in our experiments. *ID* is used to identify the instance throughout the paper. *Size* is the number of requests in the problem. *# Low* and *# High* are the number of low and high-altitude requests in each problem. *Best conflicts* and *best overlaps* are the best known values for each problem for these two objective functions. The best value for Mar07 is a new best value (the prior best was 774).

Our AFSCN dataset consists of 12 days of real data identified by their dates. Table 2 shows characteristics for these problem instances. The seven older days of data are smaller problems that are easily solved by most of the approaches we have tried. The five newer days are substantially larger problems that are more difficult.

3. NEIGHBORHOOD SEARCH

We encode potential solutions using a permutation π of the n task IDs, $[1..n]$. A *schedule builder* is used to generate solutions from the permutation. In effect, the permutation π acts as a priority queue, and the schedule builder places task requests in the schedule based on the order that they appear in π. Each task request is assigned to the first available resource from its list of alternatives and at the earliest possible starting time. This assignment treats the list of alternatives as a rank order, although the actual ordering is arbitrary.

When minimizing the number of conflicts, if the request cannot be scheduled on any of the alternative resources, it is dropped from the schedule (i.e., bumped). When minimizing the sum of overlaps, if a request cannot be scheduled without conflict on any of the alternative resources, it is placed so as to create the minimal overlap with previously scheduled requests. The last two columns of Table 2 show the best-known values for both evaluation functions.

We implemented a next-descent hill-climber that employs the *shift* operator; we accept new solutions that are better or equally good. From a current solution π, a neighborhood is defined by considering all $(n-1)^2$ pairs (x, y) of positions in π, subject to the restriction that $y \neq x-1$. The neighbor π' corresponding to the position pair (x, y) is produced by *shifting* the job at position x into position y, while leaving all other relative job orders unchanged. If $x < y$, then $\pi' = \text{SHIFT}(\pi, x, y) = (\pi(1), ..., \pi(x-1), \pi(x+1), ..., \pi(y), \pi(x), \pi(y+1), ..., \pi(n))$. If $x > y$, then $\pi' = \text{SHIFT}(\pi, x, y) = (\pi(1), ..., \pi(y-1), \pi(x), \pi(y), ..., \pi(x-1), \pi(x+1), ..., \pi(n))$. The pseudocode for the hill-climber and neighborhood variants are shown in Figure 2.

3.1 Complete Neighborhoods for AFSCN

Due to the discrete nature of the evaluation functions and the influence of the schedule builder, most of the options in the shift neighborhood are equivalent – steps on a plateau. Approximately

```
N1-HILL-CLIMBER(π)
  for num-evals from 1 to 50000
      do x ← RANDOM(N)
         π' ← π
         for y from 0 to N − 1
             do if x = y or x = y − 1
                   then continue
                τ ← SHIFT(π, x, y)
                if EVAL(τ) ≤ EVAL(π')
                   then π' ← τ
  return π'
```

```
N2-HILL-CLIMBER(π)
  for num-evals from 1 to 50000
      do x ← y ← 0
         while x = y or x = y − 1
             do x ← RANDOM(N)
                y ← RANDOM(N)
         π' ← SHIFT(π, x, y)
         if EVAL(π') ≤ EVAL(π)
           then π ← π'
  return π
```

```
RESTRICTED-HILL-CLIMBER(π, G)
  for num-evals from 1 to 50000
      do x ← y ← 0
         while x = y or x = y − 1
             do x ← RANDOM(N)
                y ← POS(π, R-ADJ(x, G))
         π' ← SHIFT(π, x, y)
         if EVAL(π') ≤ EVAL(π)
           then π ← π'
  return π
```

Figure 2. Pseudocode for each of the algorithms we use in this paper. N1-HILL-CLIMBER selects a random x position and shifts it into all other possible positions. N2-HILL-CLIMBER selects random x and y positions. RESTRICTED-HILL-CLIMBER selects a random x and chooses y from the tasks adjacent to vertex x in the interaction graph G. In this pseudocode, RANDOM(N) returns an random integer uniformly from $(0, N-1)$, R-ADJ(x, G) returns the label of a randomly selected adjacent neighbor of x in G, and POS(π, *value*) returns the position of *value* in π.

40% of the entire neighborhood results in exactly the same *schedule* (Barbulescu et al., 2004a); closer examination reveals that 60-80% are equal-valued though they are translated into different schedules. Our first intuition was to use a structured, complete neighborhood. The **N1** ordered neighborhood randomly chooses a task (i.e., a permutation position) x, and then evaluates the neighbors produced by systematically shifting x into *each* possible $n-1$ other positions; if no position is acceptable, another x is selected without replacement. Unfortunately, although systematic and easy to program, we found this neighborhood performed poorly (Barbulescu et al., 2004a). Further investigation showed a detrimental interaction between the domain and the schedule builder. When a shift produces a poorer evaluation, it usually signals that x is now blocked by the earlier task and no shift of x later in the schedule can affect that blockage, but many evaluations may be expended trying. If we count the kinds of moves seen *during* search under N1, more than 80% of the considered changes result in worse evaluations; of the remaining 20% (which constitute the actual moves taken), most are plateau moves (equivalent evaluations) and only a few are actually improving moves. This neighborhood induces a significant negative bias against improving or equal moves.

To mitigate this bias, the **N2** unrestricted neighborhood operator randomly selects both x, the task to be shifted, and y, the new position where x is to be inserted. Search algorithms that use this type of random neighborhood move are called "Stochastic Hill Climbers" because they do not systematically explore a neighborhood (Ackley, 1987). **N2** results in a major performance improvement, producing performance competitive with the best previous solutions.

3.2 Restricted Neighborhoods for AFSCN

Restricted "critical path" neighborhoods are key to achieving good performance in job-shop and flow-shop scheduling domains. Given that 40% of shifts result in no change to the schedule in AFSCN, one would expect that restricting the search neighborhood to only the tasks that induce a change would produce more efficient search.

Given a task u to be moved, the **N3** move operator restricts neighbors to only those tasks that are known to interact with u. More formally, for tasks u and v we define $interacts(u, v) = true$ if, on the same resource, $r_v \leq r_u \leq d_v$ or $r_v \leq d_u \leq d_v$, where r and d are the release and due dates, respectively. Given alternative scheduling resources, two tasks interact when they contend on one or more common resources. The "interaction" heuristic bears some resemblance to other contention measures, such as the SumHeight heuristic (Beck et al., 1997), where contention is measured across resources. Interaction uses a similar idea but expresses the pair-wise contention across tasks.

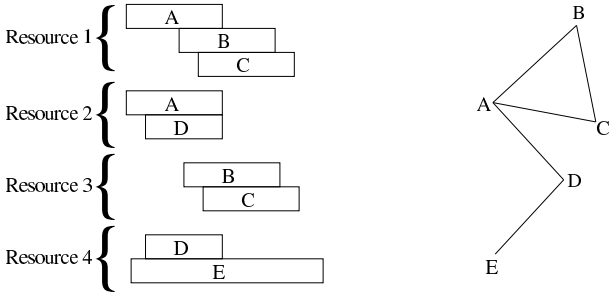

Figure 3. An example of interaction between five requests scheduled on four resources. For this idealized problem, the edge set $E = \{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{D, E\}\}$.

Task interaction can overestimate the actual amount of contention in the schedule. It includes the entire time window (from release to due date) and disregards processing time. In some situations, it may be possible to schedule both tasks within their respective time windows on the same resource. One of the tasks could also be scheduled on another alternative resource. We calculate pair-wise task interaction for all tasks to build an undirected, unweighted graph where vertices are the tasks and existing edges indicate interaction.

DEFINITION 1 *An interaction graph, G, is an undirected graph, $G = (V, E)$, where the set of vertices, V, is the set of scheduling tasks and E consists of edges between vertices $\{u, v\} \mid u, v \in V, u \neq v, interacts(u, v) = true$.*

The interaction graph is designed to provide information about the *potential* conflict between all pairs of tasks in the schedule. Figure 3 shows a simple interaction graph for an idealized oversubscribed problem. Note that interaction is not transitive; it is possible for two tasks that do not interact to both interact with a third task. This case is shown in the example, where tasks A and E both interact with D, but not with each other.

We calculate the interaction graphs for all days of data. The computational cost of calculating the interaction graph is small (less than a second). Figure 4 illustrates the largest connected component

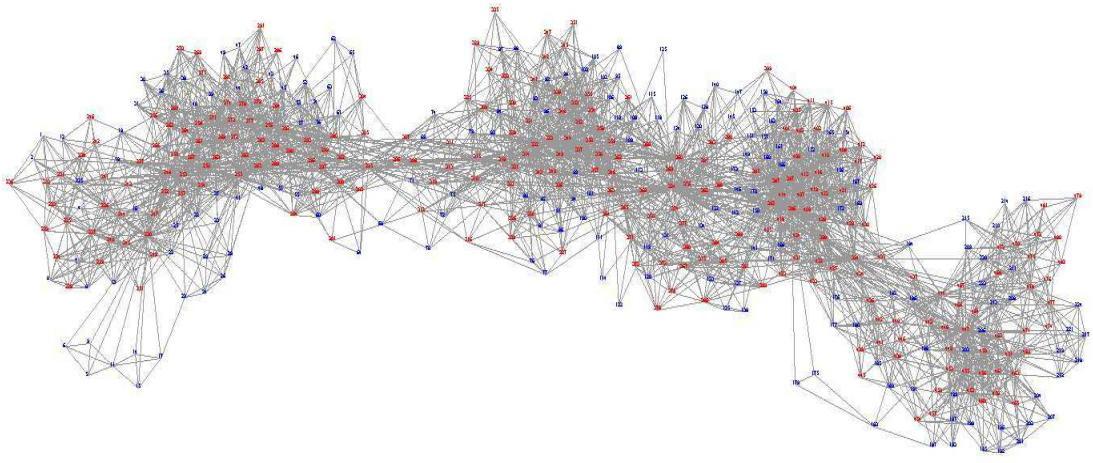

Figure 4. A force-directed layout of the largest connected component in G for Mar07. The problem contains a single connected component that spans most (92%) of the problem; the remaining tasks have zero degree. High-altitude tasks (in red) are the most connected tasks and are usually in the center. Low-altitude tasks (in blue) are less connected and tend to be along the outside of the graph.

	Tasks	Connected	Avg. Degree	L	Tasks of Zero Degree
Day1	322	295	6.01	4.61	27 (0.083)
Day2	302	273	6.31	4.61	29 (0.096)
Day3	311	281	6.10	4.60	30 (0.096)
Day4	318	289	6.20	4.57	29 (0.091)
Day5	305	274	6.20	4.61	31 (0.102)
Day6	299	274	6.05	4.65	25 (0.084)
Day7	297	271	6.10	7.01	26 (0.088)
March0702	483	440	8.32	3.80	43 (0.089)
March2002	457	426	8.78	3.67	31 (0.068)
March2603	426	396	7.38	4.02	30 (0.070)
April0203	431	396	7.14	4.96	35 (0.081)
May0203	419	388	7.09	5.03	31 (0.074)

Table 3. Characterization of G for each day of AFSCN data. The first column shows the number of tasks in the problem; the second shows the number of tasks connected to the largest connected component. The next two columns show the average degree and the average path length, L. The last shows the number of zero degree tasks in each problem. The number in parenthesis is the ratio of zero degree tasks in the problem.

of G for Mar07; this component includes 92% of the vertices. Tasks are roughly in order by ID from left-to-right. Table 3 shows a summary of the largest connected component for all the days of data. The connectedness of these graphs is sparse ($|E| \ll \binom{n}{2} < \mathcal{O}(n^2)$) and shows a low average path length between any two tasks.

The **N3** move operator uses G to restrict the neighborhood explored by local search; the goal of using **N3** is to focus search on those adjacent new solutions that force a change in the schedule. We iteratively choose a random position x and shift the task in that position to the position of a randomly selected, interacting neighbor. **N3** dramatically reduces the neighborhood size from $\mathcal{O}(n^2)$ to the average degree per vertex (see Table 3, column 'Avg. Degree'). So **N3** also reduces the expected number of neighbors evaluated before selecting a move.

To control for the effects of the structure in the neighborhood, we also implemented a fourth neighborhood: a random, restricted neighborhood. The **N4** move operator creates a random graph with the same degree per vertex as the interaction graph. Each edge in the random restricted graph is randomly connected to another randomly chosen task (excluding itself). **N4** shifts the task in position x to a randomly selected neighbor in the graph.

4. EXPERIMENTS

We first compare the performance of the four neighborhoods. Table 4 shows the final evaluation distributions for 50,000 evaluations over 90 runs. N1 reaches the best-known values much less frequently than the other three neighborhoods, while the other neighborhoods appear equivalent. We focus our analysis on the differences found between the unrestricted and restricted neighborhoods.

Although the minimum values of N2, N3 and N4 are identical, the means and standard deviations vary. Table 4 lists the p-values for the N2 (unrestricted) neighborhood compared to each of the other three neighborhoods using a one-tailed t-test. In nearly all cases, the N2 distributions have lower means and are significantly different than the distributions of the results of other neighborhoods. Thus, the restricted neighborhoods hurt more than they help search for AFSCN.

In addition, informed restriction (N3) does not dominate random restriction (N4). Using a t-test ($\alpha < .05$), N4 is significantly better on Day1, Day2, Day3, and Day5 in both evaluations; it is also better on Apr02 in minimizing conflicts.

	N1				N2			N3				N4			
	p	min	μ	σ	min	μ	σ	p	min	μ	σ	p	min	μ	σ
Day1	-	9	11.07	1.32	8	8.06	0.23	-	8	9.64	0.93	-	8	9.19	0.98
Day2	-	4	5.03	1.02	4	4.00	0.00	-	4	4.86	0.83	-	4	4.51	0.67
Day3	-	4	6.74	1.47	3	3.00	0.00	-	3	3.36	0.61	-	3	3.18	0.41
Day4	-	4	6.99	1.55	2	2.00	0.00	-	2	2.80	0.62	-	2	2.98	0.65
Day5	-	4	6.90	1.23	4	4.09	0.29	-	4	5.42	0.81	-	4	5.11	0.76
Day6	-	6	9.70	1.67	6	6.00	0.00	-	6	6.29	0.46	-	6	6.46	0.64
Day7	-	6	7.91	1.12	6	6.00	0.00	-	6	6.44	0.66	-	6	6.31	0.51
Mar07	-	53	58.02	2.55	42	42.04	0.21	-	42	42.90	0.75	-	42	42.87	0.78
Mar20	-	32	40.27	2.70	29	29.01	0.11	-	29	29.13	0.34	-	29	29.29	0.46
Mar26	-	21	25.94	2.14	17	17.12	0.33	.77	17	17.09	0.29	.02	17	17.24	0.43
Apr02	-	32	36.28	2.17	28	28.00	0.00	-	28	28.87	1.00	-	28	28.59	0.86
May02	-	14	16.79	1.49	12	12.00	0.00	ns	12	12.00	0.00	*	12	12.08	0.27

	N1				N2			N3				N4			
	p	min	μ	σ	min	μ	σ	p	min	μ	σ	p	min	μ	σ
Day1	-	104	172.18	33.29	104	105.73	1.49	-	104	120.30	15.76	-	104	112.49	8.52
Day2	-	13	36.57	17.20	13	13.00	0.00	-	13	29.72	16.63	-	13	21.80	10.91
Day3	-	35	82.22	24.86	28	28.00	0.00	-	28	31.37	6.77	-	28	29.43	4.17
Day4	-	19	64.82	27.89	9	9.13	0.72	-	9	20.67	11.95	-	9	20.64	10.43
Day5	-	31	65.41	22.00	30	30.01	0.11	-	30	47.13	13.65	-	30	41.23	10.59
Day6	-	50	96.98	27.60	45	45.00	0.00	-	45	49.03	8.83	-	45	47.31	6.58
Day7	-	49	87.59	25.38	46	46.00	0.00	-	46	47.90	5.14	*	46	47.16	3.77
Mar07	-	1173	1364.28	89.46	773	778.59	7.64	-	773	788.43	13.30	-	773	787.81	14.27
Mar20	-	697	852.50	73.47	486	495.08	6.32	-	486	501.36	12.51	-	486	501.33	13.07
Mar26	-	425	624.23	82.52	250	258.96	25.67	.38	250	260.14	25.50	.07	250	266.22	39.11
Apr02	-	958	995.26	68.72	725	731.42	13.76	-	725	754.80	29.63	-	725	756.96	22.54
May02	-	170	243.76	28.71	146	146.00	0.00	ns	146	146.00	0.00	*	146	146.33	1.25

Table 4. Summary statistics for the final evaluation distributions of conflicts (upper table) and overlaps (lower table). These statistics are taken over 90 runs of 50,000 evaluations each. P-values are computed between unrestricted search and restricted search using a one-tailed t-test that one distribution is lower. High values are written as numbers. Otherwise, a dash indicates significance at the $\alpha < .0001$ level; a star indicates significance at the $\alpha < .01$ level; insignificance is marked by 'ns'.

We conjecture that unrestricted search (N2) converges to the best known values more frequently than restricted search (N3 and N4). To judge this hypothesis, we counted the number of converging and non-converging runs (out of 90) of these three neighborhoods for each day of data. We then performed a χ^2 test that the proportion of converging runs was the same for N2 as compared to N3 and N4. Mar26 and May02 had similar counts, so the test was not significant. For the other ten days of data, this test revealed that the success of convergence significantly depends on the neighborhood ($p < .01$).

One hypothesis for such variance in the performance of these algorithms is that non-converging runs get stuck on large, suboptimal basins. If this were true, one might expect the final evaluations to be distributed somewhat uniformly above the best-known values. We examined histograms of the final evaluations over 90 runs and found that the non-converging runs end close to the best known values. N2 almost always gets more runs closer, but the difference is still small. For minimizing conflicts, N2 usually gets within one conflict while N3 usually gets within three conflicts. For overlaps, there is slightly more complex behavior. On the seven older days of data, N2 finds solutions within one or two units of overlap while N3 usually gets within 100. On the five new days of data, N2 and N3 closely mimic each others' final evaluations. Most runs reach within 100 of the best known values.

To assess local differences in neighborhoods, we also examined the number of improving, non-improving, and equal moves under each neighborhood. For the improving moves, we also histogram the change in evaluation. We attempted to correlate these changes in evaluation with specific tasks or task attributes, but found little correspondence of move quality with problem specific information. These results, coupled with the lack of competitive advantage for N3 over N4, lead us to the final conclusion

in our examination of our structured restricted neighborhood: the structured interaction graph provides little advantage over a randomly selected restriction. These graphs do reduce the neighborhood but still remain connected enough such that they can find reasonable solutions.

5. SUMMARY AND FUTURE WORK

We examined the effects of problem motivated structure and restricted neighborhood size on the performance of neighborhood operators for a real world scheduling application, AFSCN. Following conventional wisdom, we hypothesized that we could reduce the neighborhood using problem specific structure in a restricted neighborhood. The result was somewhat surprising in that this significantly degraded performance (according to a one-tailed t-test). Search using a restricted neighborhood converges to the best-known values less frequently and shows no major improvement in taking steps that change the evaluation any more than unrestricted search. Moreover, randomly restricted search significantly outperforms structured restricted search for almost half of the problems.

For AFSCN, a restricted neighborhood markedly under-performs an unordered, full neighborhood in next-descent local search. Our evidence suggests that the search is a random walk. We conjecture that search can be modeled as a Markov chain, and we are currently developing a model of the shift neighborhood for unrestricted search. Preliminary results indicate that this model may be quite accurate for AFSCN. We are also extending these analyses to another oversubscribed scheduling domain.

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.

We thank Dr. James T. Moore, Associate Professor, Dept. of Operational Sciences, Air Force Institute of Technology and Brian Bayless and William Szary from Schriever Air Force Base for providing the data.

We also thank the anonymous reviewers for their time in critiquing this work and offering suggestions for improvement.

References

Ackley, David (1987). A Connectionist Machine for Genetic Hillclimbing. Kluwer Academic Publishers.

Barbulescu, L., Howe, A., Whitley, L., and Roberts, M. (2004a). Trading places: How to schedule more in a multi-resource oversubscribed scheduling problem. In *International Conference on Automated Planning and Scheduling (ICAPS-04)*.

Barbulescu, L., Watson, J., Whitley, D., and Howe, A. (2004b,January). Scheduling space-ground communications for the Air Force satellite control network. *Journal of Scheduling*, 7(1).

Beck, J.C., Davenport, A.J., Sitarski, E.M., and Fox, M.S. (1997). Texture-based Heuristic for Scheduling Revisited In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. Providence, RI: AAAI Press, 241–248.

Joslin, D. E. and Clements, D. P. (1999). “Squeaky Wheel" optimization. In *Journal of Artificial Intelligence Research*, 10, 353–373.

Pinedo, M. (2002). Scheduling: Theory, Algorithms, and Systems. Upper Saddle River, NJ: Prentice Hall.

Whitley, D., Rana, S., and Heckendorn, R. B. (1997). Representation Issues in Neighborhood Search and Evolutionary Algorithms. In D. Quagliarella and J. Periaux and C. Poloni and G. Winter (Eds.), *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science* (pp. 39-57). New York: Wiley.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.