Optimizing Observation Scheduling Objectives

John L. Bresina Recom Technologies
Robert A. Morris Florida Institute of Technology
William R. Edgington Recom Technologies

NASA Ames Research Center, Mail Stop: 269-2
Moffett Field, CA 94035-1000 USA

{bresina, morris, wedgingt}@ptolemy.arc.nasa.gov

Abstract

In this paper, we present a novel approach that enables the automatic generation of high quality schedules with respect to a given objective function. The approach involves the combination of two techniques: GENH, which automatically generates a search heuristic specialized to the given problem instance, and HBSS, which employs the heuristic as a bias within a stochastic sampling method. We empirically investigate the performance of these techniques, individually and in combination, within a real-world application of observation scheduling.

Introduction

In this paper, we present a novel approach that enables the automatic generation of high quality schedules with respect to a given objective function. The approach involves a combination of two techniques: GENH [7], which automatically generates a search heuristic specialized to the given problem instance, and *Heuristic-Biased Stochastic Sampling* (HBSS) [2], which employs a given heuristic as a bias within a stochastic sampling method. These approaches have been implemented within the Associate Principal Astronomer (APA) system, which provides management support and improved observation scheduling for fully automatic, terrestrial telescopes.

Deriving search heuristics that are both accurate and computationally inexpensive is a difficult endeavor for most problems. This is especially true when not just any solution is acceptable and the heuristic is further required to find a high quality solution. Furthermore, the larger the class of problem instances, the more difficult it is for a search heuristic to perform consistently well over the class.

GENH was initially motivated by the idea that the role of a domain expert should be limited to constructing the domain-specific objective function. Before GENH, the search heuristic had to be manually tuned using a "generate and test" exploration, which was a time-consuming and difficult process that no one wanted to repeat very often. However, within this application domain, the nature of the scheduling problem changes day-to-day due to the changing position of the celestial targets as well as the addition or modification of the observation requests. Hence, the search heuristic that performs well for one day's scheduling problem may not do so well for the next one. Since it is difficult to find a single heuristic that performs well across the wide range of problem instances, the automation of this task by GENH is important for this scheduling application.

The number of feasible schedules of telescope observations for a given day is on the order of 10^{100}; given such large search spaces, it is very unlikely that any local heuristic can yield a globally optimal schedule via greedy search. The underlying assumption behind the HBSS approach is that strictly adhering to a search heuristic often does not yield the best solution and, therefore, that exploration off the heuristic path can prove fruitful. Within HBSS, the balance between heuristic adherence and exploration is controllable by the user. The accuracy of the search heuristic is an important factor in choosing the appropriate balance to use; typically, the less accurate the heuristic, the weaker the bias towards heuristic adherence should be. Another important factor is the amount of solution generation time available; if there is not much time available, then exploration must be limited and a stronger bias is recommended.

The outline of the rest of the paper is as follows. We first briefly present background on the application domain, and we next present the GENH and HBSS techniques. Then we empirically demonstrate the improvement yielded by the two techniques, individually and in combination. The improvement is with respect to the original dispatch scheduling technique encoded in the telescope controller. In the last section, we offer some concluding remarks.

Observation Scheduling Domain

The input to the APA observation scheduler is a set of requests, expressed using the Automatic Telescope Instruction Set, or ATIS[1]. Each request is composed of a sequence of telescope movements and instrument commands, as well as scheduling constraints and preferences.

A request is *enabled* on a given night if all of its constraints are satisfied. The primary constraint is that each request can be executed only within a specific time

interval. On a given night, an observation can only be executed during the intersection of the observation's interval and that night's interval of darkness. Enablement is also affected by the moon – each observation includes a constraint regarding whether the moon must be up, down, or either. Furthermore, even those observations that are enabled when the moon is up cannot be executed when its observation targets are too close to the moon. We refer to the time interval (for a given night) during which an observation can begin execution as its *enablement interval.* The primary preference specified is the relative priority that an astronomer assigns to each observation request. On many nights, not all of the possible requests can be executed; these relative priorities help determine which subset to execute on a given night. Each observation typically takes around five minutes to execute, and between 60 and 140 observations can be executed during a night.

We formulate this observation scheduling problem as a *refinement search* (also called *constructive search*). The scheduler's search space is organized chronologically as a tree, where the root node consists of the world model state at the beginning of the night. Each arc out of a search tree node represents an enabled request. The task of the APA scheduler is to find a sequence of observations that achieves a good score according to the objective function. For further domain details, see [3; 5].

The ATIS standard also specifies an heuristic dispatch policy which can be used to select the next observation to execute. The policy is expressed as four selection rules: *priority*, *number-of-observations*, *nearest-to-end-window*, and *file-position*. When the controller has finished executing an observation, it first determines the currently enabled observations and then applies these four rules in the sequence given to select one. Each rule is used to break ties that remain from the application of those that preceded it. If the result of applying any rule is that there is only one observation remaining, that observation is selected for execution and no further rules are applied. Since there can be no file-position ties, the dispatch policy is deterministic.

ATIS dispatch is a robust scheduling method that has been used fairly successfully for several years to schedule automatic photoelectric telescopes at Fairborn Observatory before the development of the APA. (See [6] for a performance evaluation of ATIS dispatch.) The dispatch decisions are determined purely locally, without look-ahead; by contrast, the APA uses a search-based scheduler (for APA scheduler details, see [4]).

The GenH Technique

In this section, we describe the GENH technique, first introduced in [7]. GENH automatically generates a search heuristic that is specialized for the given problem instance. For observation scheduling, the input to GENH consists of an objective function, a set of candidate heuristic attributes, and the set of observation requests. The output from GENH is the best multi-attribute heuristic found as the result of a search through

```
GenH(objective, attributes, requests)
begin
{generate seed}
bestH = random_seed(attributes)
best_score = eval(bestH)
{initialize boundary variables}
boundary = 0.5
granularity = 0.1
{repeat until granularity minimum exceeded}
repeat until granularity < 0.004
    {tune each attribute separately}
    for each A ∈ random_order(attributes)
        candidates = neighbors(bestH, A,
                               granularity, boundary)
        if (∃ H ∈ candidates such that
            eval(H) < best_score)
        then update best_H and best_score
    endfor
    {reset boundary variables}
    boundary = boundary / 5.0
    granularity = granularity / 5.0
endrepeat
return bestH
end
```

Figure 1: The version of the GENH heuristic generation algorithm employed in the reported experiments. The function `eval` invokes the scheduler and applies the objective function to the resulting greedy solution.

a space of candidate heuristics. GENH solves an optimization problem, defined as follows. The candidate solution set, $\mathcal{S}$, is $\{< w_i > \mid \forall_i w_i \in [0,1]$ and $\Sigma_i w_i = 1\}$; *i.e.*, the set of all weight vectors such that each weight is in the interval $[0,1]$ and the sum of the weights equals 1. GENH's evaluation of a candidate heuristic is the objective function score of the schedule found *via* greedy search using that candidate.

GENH uses *local search* (also called *repair search*) to conduct a search through $\mathcal{S}$. In a local search space, each node corresponds to a solution candidate, and for each node, a *neighborhood function* defines a set of neighbors, each of which corresponds to a local modification of the candidate solution. A simple local search process, referred to as *iterative improvement*, starts at a randomly generated "seed" and, at each decisions point, chooses the neighbor of the current node that most improves the current evaluation score.

Many variants of local search techniques were tried within GENH. We present here a simplified version that was used in the reported experiment; this specific algorithm is shown in Figure 1. First, GENH generates an initial seed by randomly selecting values for each weight (w_i) and then normalizing them to sum to 1. This heuristic is then evaluated. GENH's evaluation of a candidate heuristic consists of invoking the APA scheduler on the problem instance to perform greedy search using the candidate heuristic, and then scoring the resulting schedule

with the given objective function (accomplished by the `eval` function in the figure).

In GENH's search space, the neighbors of a node are generated by adjusting the weight of a selected single attribute and then dividing each weight by the sum of the weights so that the resulting vector sums to 1 (accomplished by the `neighbors` function in the figure). The best neighbor resulting from tuning one attribute is the starting point for the adjustments made to the next selected attribute, until all the attributes have been singly tuned. In the reported experiment, the attributes are processed in a random order.

The set of adjustments to the selected weight is based on two parameters: a *boundary* parameter which determines the maximum adjustment (increment or decrement) to the weight's current value, and a *granularity* parameter which determines the set of adjustments (uniformly distributed) between these bounds. After each attribute has been tuned once, the granularity and boundary parameters are decreased, and the entire set of attributes is tuned again. On each iteration a smaller neighborhood of the current best heuristic is searched with a finer granularity. The search terminates when the granularity parameter exceeds user-specified minimum.

In the reported experiment, the granularity parameter is initialized to 0.1 and the boundary parameter is initialized to 0.5. If the initial value for the weight selected for tuning is v, the set of neighbor values considered is $\{v_n = \max(0, v - 0.5) + k \times 0.1 \mid v_n \leq \min(1, v + 0.5)\}$, where k is an integer. After each iteration, both parameters are divided by 5; hence, there is a constant upper bound on the number of neighbors. Thus in the second iteration, the set of neighbor values is $\{v_n = \max(0, v-0.1)+k \times 0.02 \mid v_n \leq \min(1, v+0.1)\}$. In the reported experiment, the search was terminated when the granularity parameter is less than 0.004 (*i.e.*, after three iterations of the repeat loop).

The HBSS Technique

In this section, we describe the *Heuristic-Biased Stochastic Sampling* (HBSS) algorithm, first introduced in [2]. Within HBSS, the desired balance between heuristic adherence and exploration in the search space is determined by specifying a *bias function* and a *ranking function*. Both of these functions enable the user to encode information about the heuristic function; this additional knowledge is employed during search to tailor the guidance provided by the heuristic.

The ranking function enables the encoding of information regarding what relevant distinctions to make within the range of the heuristic function. The ranking function partitions the heuristic's range into equivalence classes and determines the magnitude of the quality differences between classes. The bias function enables the encoding of information regarding the accuracy of the heuristic and the amount of exploration that is desired. A stronger bias tends to follow the heuristic's advice more often and a weaker bias tends to explore farther off the greedy trajectory in the search tree.

HBSS-iterate(root, heuristic_fnc, rank_fnc, bias_fnc)
begin
current_state = root
nodes = **successors**(root)
repeat until empty(nodes)
    current_state = **HBSS-select**(nodes,
            heuristic_fnc, rank_fnc, bias_fnc)
    nodes = **successors**(current_state)
endrepeat
return current_state
end

HBSS-select(nodes, heuristic_fnc, rank_fnc, bias_fnc)
begin
score nodes with heuristic_fnc
rank nodes with rank_fnc based on scores
weight nodes with bias_fnc based on ranks
normalize weights to yield *selection probabilities*
select node *stochastically* according to probabilities
return selected node
end

Figure 2: `HBSS-iterate` performs one iteration of the HBSS algorithm, and `HBSS-select` is used at each decision point within `HBSS-iterate`; where **successors** returns the states resulting from application of each observation enabled in the given state.

The algorithm for generating one sample, or schedule, is given in Figure 2; this version assumes that the search tree has finite branching and finite depth, as is the case for our domain. At each decision point, the alternative choices are scored according to the given heuristic function, and each choice is assigned a rank, according to the given ranking function, based on these scores. We assume that ranks are positive integers and that the top rank is 1. The given bias function is then used to assign a non-negative real-valued weight to each choice based on its rank. The assigned weights are then normalized by dividing each one by the sum of the weights. The normalized weight for an alternative choice represents its probability of being selected; a choice is selected according to these probabilities by a weighted stochastic process.

A typical ranking function is one that puts choices with equal heuristic scores into the same equivalence class and then assigns these classes consecutive integers. A typical family of bias functions are the *polynomial bias functions* defined as $\texttt{poly}_n(r) = r^{-n}$, where r is the rank and n is the polynomial degree.

Figure 3 illustrates how `HBSS-select` works. In this hypothetical example, at each decision point there are three choices which are assigned unique ranks. The table in the figure shows how the three heuristic scores for an example decision point are converted into three selection probabilities by the steps in `HBSS-select` using a $\texttt{poly}_1$ bias function; in this example, lower scores are better. Each node in the tree is labeled with its assigned

Heuristic:	100.25	150.67	300.04
Rank:	1	2	3
Bias ($1/r$):	1.00	0.50	0.33
Probability:	0.55	0.27	0.18

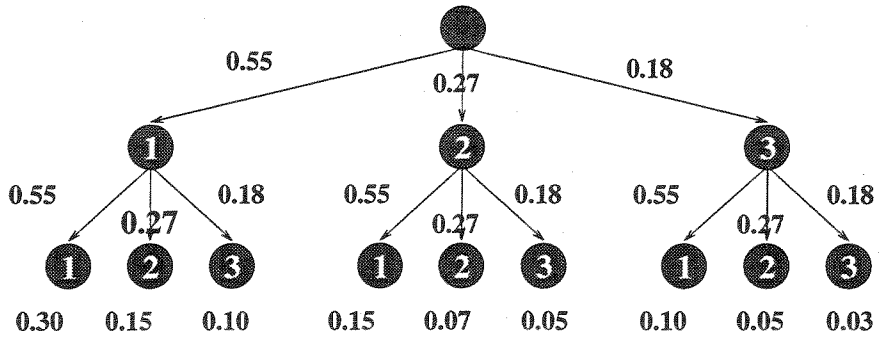

Figure 3: Example of solution probability distribution induced by `HBSS-iterate` with a $\texttt{poly}_1$ bias of $1/r$ on an hypothetical tree. In the tree, nodes are labeled with the assigned rank, arcs are labeled with the selection probability, and the bottom numbers indicate the probability of reaching the corresponding leaf node.

rank and each arc is labeled with the resulting selection probability. The number below each leaf node of the tree is the probability of reaching that leaf, which is the product of the selection probabilities of all the arcs in the path from the root node to the leaf node. Hence, the incremental application of `HBSS-select` induces a probability distribution over the space of possible solutions.

Typically, the `HBSS-iterate` routine is executed some number of times, based on the available schedule generation time, and the best schedule with respect to the given objective function found is returned. It is difficult to predict which bias function is going to yield the best performance on a given problem, with a given heuristic and a given number of samples. One approach to ameliorating this problem is to use more than one bias function and alternate on each sample. This would allot an equal number of samples (within 1) to each bias function regardless of the number of total samples. Within the APA scheduler, a "sample proportion" can be specified for each bias function so that uneven computation shares can be allotted. In the reported experiment, we use this multi-bias HBSS approach with $\texttt{poly}_2$, $\texttt{poly}_3$, and $\texttt{poly}_4$, each getting an equal share of samples.

Experiment Methodology

In this section, we describe the methodology of the computational experiments, which were carried out within the observation scheduling problem class. One of the experiment's goals is to test the hypothesis that adaptive, stochastic-based scheduling significantly improves the quality of the schedules over a non-adaptive, deterministic approach in the telescope observation domain. A second goal is to empirically measure the share of the improvement yielded by GENH and HBSS individually, as well as to ascertain whether their combined use is better than either single technique.

To test our hypothesis, one of the scheduling methods employed is a non-adaptive, deterministic technique based on ATIS dispatch (defined in the second section). The dispatch policy was simulated as greedy search with the following "dispatch heuristic": $0.625 \times \texttt{priority} + 0.3125 \times \texttt{run count} + 0.0625 \times \texttt{enablement time}$. The attribute *run count* is the number of times the observation occurs in the current schedule prefix, and the attribute *enablement time* is the time currently remaining in the observation's enablement interval. Lower priority numbers indicate more importance, and lower values are also preferred for the run count and enablement time attributes; hence, a lower heuristic score is better.

For the test cases, we selected a set of problem instances over the 251 day interval of Julian Dates $[2450400, 2450650]$. This range of instances assures variation in problem characteristics (specifically, with respect to load capacity and phase of the moon). The objective function is the weighted summation of two attributes: one which penalizes enabled observations that are missed in the schedule based on their priority, and one which prefers schedules that minimize the average airmass of the scheduled observations. The priority penalty is computed as $\sum 10^{Pmax-P(mo)}$, where the summation is over all missed observations mo, $P(mo)$ is the priority of observation mo, and $Pmax$ is the maximum priority in the observation set. The average airmass is calculated as $\sum A(so) \times D(so) \;/\; (dawn - dusk)$, where the summations are over all scheduled observations so, $A(so)$ is the airmass of so, and $D(so)$ is the duration of so. Airmass is computed based on the secant of the telescope pointing angle.

The weights assigned in the objective function are 100 for the priority penalty and 1 for the average airmass. Given the ranges of these two attributes, the attribute weights ensure that if one schedule has a better score

with respect to the priority penalty attribute than another, then it will have a better objective function score regardless of the average airmass score; hence, the airmass attribute will only break ties between schedules that have the same priority penalty score.

The set of candidate attributes for GENH includes the attributes contained in the dispatch heuristic, as well as an additional attribute, *airmass*. The reason for including the additional attribute is to exploit GENH's ability to examine arbitrary collections of attributes and to weed out noneffective ones.

We carried out a comparative analysis of the following four scheduling techniques:

- *Greedy Dispatch*: Deterministic greedy search using the dispatch heuristic
- *Greedy GenH*: Deterministic greedy search using the problem's GENH heuristic
- *HBSS Dispatch*: Multi-bias HBSS using the dispatch heuristic
- *HBSS GenH*: Multi-bias HBSS using the problem's GENH heuristic

The comparisons of interest are the following: (*i*) Greedy GENH *vs.* Greedy Dispatch, (*ii*) HBSS Dispatch *vs.* Greedy Dispatch, (*iii*) HBSS GENH *vs.* Greedy GENH, and (*iv*) HBSS GENH *vs.* HBSS Dispatch.

On each problem instance in the test suite, the performance of each of the four techniques was evaluated as follows. For the two deterministic techniques, the performance evaluation is the objective function score of the single schedule generated. For the two stochastic sampling methods, five runs of 15 samples each were performed. Hence, the $\texttt{poly}_2$, $\texttt{poly}_3$, and $\texttt{poly}_4$ bias functions are each employed on 5 samples. For each run, the best objective function score from the 15 samples is collected; the performance evaluation is the average of these five best scores.

Interpretation of Results

In this section, we discuss the empirical results gathered thus far in our ongoing investigation. The performance results for the four above comparisons are illustrated in Figures 4 and 5. Rather than illustrating the comparisons in terms of the objective function scores, they are illustrated with respect to each of the unweighted attribute scores, in order to more closely examine the differences between the four techniques. Since the weighted priority penalty always dominates the weighted average airmass, the performance w.r.t. the objective function score exactly mirrors the performance w.r.t. the unweighted priority penalty. Figure 4 illustrates improvement w.r.t. priority penalty, and Figure 5 illustrates the improvement w.r.t. average airmass. Since lower scores are preferred, improvement of technique *A over* technique *B* is computed as *B*'s score − *A*'s score, and a negative improvement indicates the improvement of *B over A*. Negative improvement is utilized in Figure 5; however, since the `y-axes` in Figure 4 are logarithmic, only positive improvement is plotted.

The two plots in the first (top) row of Figure 4 show that for all but 3 of the 251 problem instances, the heuristic generated by GENH significantly outperformed the dispatch heuristic when both were used in greedy search. For the three exceptions (JDs 2450480, 2450481, and 2450543), either an equally good (or better) heuristic did not exist in GENH's search space $\mathcal{S}$, or else GENH failed to find one within the limitations of the specific search strategy employed.

The two plots in the figure's second row show that, on every problem instance, HBSS with the dispatch heuristic was able, in only 15 samples, to find a significantly better schedule than the greedy solution. Comparing the first two rows indicates that both HBSS and GENH individually almost always achieve the same degree of improvement over greedy dispatch.

The bottom two rows in the figure illustrate how the combination of HBSS and GENH compares to each technique individually. The results plotted in the third row illustrate how much HBSS helps GENH, and the results plotted in the fourth row illustrate how much GENH helps HBSS. The right plot in the third row shows that on 5 problems, HBSS was not able to find, within 15 samples, a schedule as good as the one GENH found. The left plot in that row shows that HBSS did not often help GENH – it did so on 13 of the 251 problems; however, it significantly helped on the three problems for which GENH did much worse than greedy dispatch. This is an indication that HBSS and GENH are complementary.

The results plotted in the last row indicate that GENH helped HBSS on 87 problems, hurt HBSS on 12 problems, and had no affect on the remaining 152 problems. This suggests that giving HBSS a head start with a better heuristic can help it find better schedules, but that HBSS' performance is not strongly dependent on the quality of the heuristic – which was one of the intended features of HBSS' design! We expect that the greater the number of samples used by HBSS, the less sensitive it will be to heuristic inaccuracy.

We now turn our attention to the secondary scheduling objective: minimizing airmass. The four comparative analyses with respect to the average airmass attribute are shown in Figure 5. The two plots in the figure's top row illustrate the performance of HBSS and GENH individually, and the two plot in the bottom row illustrate the performance of the two techniques in combination.

The right plot in the top row indicates that HBSS gains the improvement over greedy dispatch with respect to the priority penalty at the cost of increased airmass. In contrast, GENH makes this tradeoff much less often and, furthermore, GENH usually gains improvement over greedy dispatch as well.

The left plot of the figure's bottom row shows that HBSS, with 15 samples, yields only minor airmass improvement over greedy search when both use the GENH heuristics. However, the right plot in that row indicates that GENH's airmass improvement transfers to the HBSS context, greatly improving HBSS with respect to this secondary attribute. These results, conjoined with the re-

sults (shown in the bottom row of Figure 4) that GENH can sometimes help improve HBSS' priority scores, indicate that HBSS and GENH are complementary.

Figure 6 illustrates GENH's output, *i.e.*, the heuristic attribute weights, over the test problem class. The sparseness of the plots for the heuristic attributes for priority and enablement time indicate that often GENH chose a zero weight, thus eliminating the attribute from the heuristic. Elimination from the heuristic happened on 173 of the 251 problems for the priority attribute and on 162 problems for enablement time; whereas it only happened twice for the airmass attribute and only once for run count. The average weights for the four attributes are as follows: airmass: 0.595, run count: 0.296, priority: 0.061, and enablement time: 0.048.

These results help explain why GENH does so well with respect to the airmass objective; however, they are very surprising with regards to the primary objective of priority. How is GENH able to improve over dispatch with respect to the priority objective when dispatch assigns the priority attribute the highest weight and GENH places such little importance on it? Recall that the priority objective attribute differs from the priority heuristic attribute. The latter can be easily evaluated on either a complete or partial schedule since it is defined in terms of what is included in the schedule. Whereas the former is the more global property of what is missing from the (complete) schedule. It is possible, due to interactions among the attributes and specifics of the observation request set, that a scheduler which locally optimizes with respect to airmass generates schedules that also score well with respect to the global priority objective.

Conclusion

In this paper, we described two approaches to improving schedule generation with respect to the domain-specific objectives. GENH improves schedule quality by improving the search heuristic, and HBSS improves schedule quality by making better use of a given search heuristic. The results demonstrate the superiority of an adaptive scheduling approach and suggest that HBSS and GENH are complementary in the following respects:

- HBSS can compensate when GENH misses the mark and produces an heuristic worse than dispatch.
- GENH enables HBSS to improve the priority score over dispatch without sacrificing airmass and, thus, to optimize both the primary and secondary scheduling objectives.

When employing HBSS in practise, the intent is to first perform a greedy search and then to run HBSS for the remaining available solution time in an attempt to find a better solution *via* informed exploration. In addition, within the APA scheduler we can use any combination of the four solution strategies discussed above; our current meta-strategy is as follows: first, greedy dispatch, second, greedy GENH, and third, n samples of HBSS GENH for the remaining allotted solution time.

We intend to explore other search strategies within GENH, *e.g.*, a multistart approach in which multiple GENH iterations are run, each with a different random seed. We also intend to investigate how the number of HBSS samples affects the relationship between GENH and HBSS. Furthermore, we have begun to address multi-night objectives, *e.g.*, resource allocation fairness, and the leverage over dispatch achieved by the combination of HBSS and GENH should become even more significant.

Acknowledgments

Thanks to Stuart Rodgers, who played a major role in the original development of GENH. Thanks also to the APA project team's former members: Mark Drummond, Keith Swanson, and Ellen Drascher, and our collaborators: Gregory Henry (Tennessee State University), Donald Epand (Fairborn Observatory), and Louis Boyd (Fairborn Observatory).

References

[1] Boyd, L., Epand D., Bresina J., Drummond M., Swanson K., Crawford D., Genet D., Genet R., Henry G., McCook G., Neely W., Schmidtke P., Smith D., and Trublood M. 1993. Automatic Telescope Instruction Set 1993. In *I.A.P.P.P. Comm.*, No. 52. Oswalt (ed).

[2] Bresina, John L. Heuristic-Biased Stochastic Scheduling. In *Proceedings of AAAI-96*, Portland, OR.

[3] Bresina J., Drummond M., Swanson K., and Edgington W. 1994. Automated Management and Scheduling of Remote Automatic Telescopes. In *Optical Astronomy from the Earth and Moon*. ASP Conference Series, Vol. 55. D.M. Pyper and R.J. Angione (eds.).

[4] Bresina, J., Edgington, W., Swanson, K., and Drummond, M. 1996. Operational Closed-Loop Observation Scheduling and Execution. In *Working Notes of the AAAI Fall Symposium, Plan Execution: Problems and Issues*. Cambridge, MA. Available in *AAAI Technical Report FS-96-01*.

[5] Drummond, M., Bresina, J., Edgington, W., Swanson, K., Henry, G., and Drascher, E. 1995. Flexible Scheduling of Automatic Telescopes over the Internet. In *Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy*. ASP Conference Series, Vol. 79. G.W. Henry and J.A. Eaton (eds).

[6] Henry, G.W., 1996. ATIS dispatch scheduling of robotic telescopes. In *New Observing Modes for the Next Century*. ASP Conference Series, Vol. 87. T. Boroson, J. Davies, and I. Robson (eds).

[7] Morris, R.A., Bresina, J.L., and Rodgers, S.M. 1997. Automatic Generation of Heuristics for Scheduling. In *Proceedings of IJCAI-97*. Nagoya, Aichi, Japan.

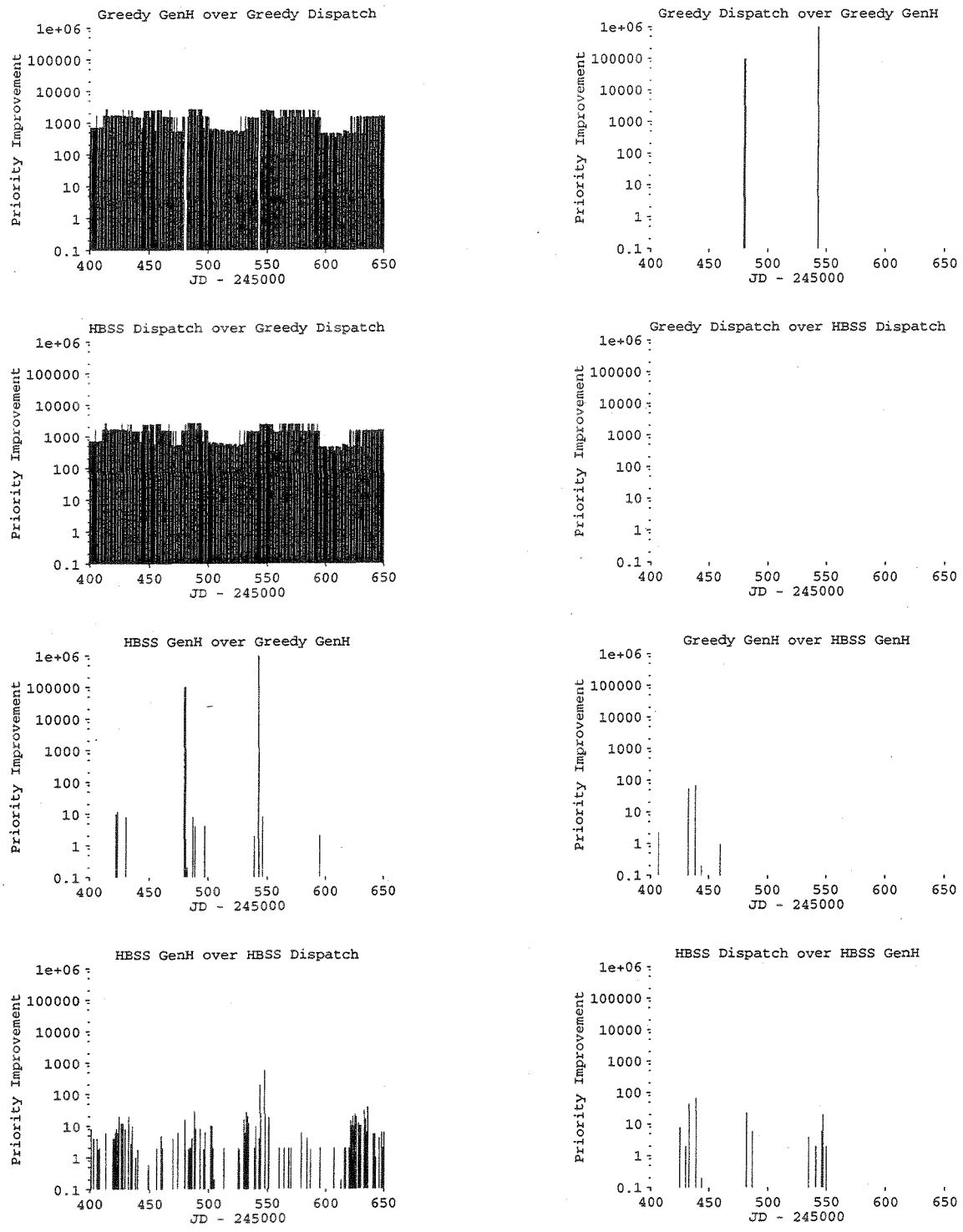

Figure 4: Improvement with respect to the priority attribute score; HBSS performance is based on the best score after 15 samples averaged over 5 runs. Note, the `y-axis` is log scale and, hence, only positive improvement is plotted.

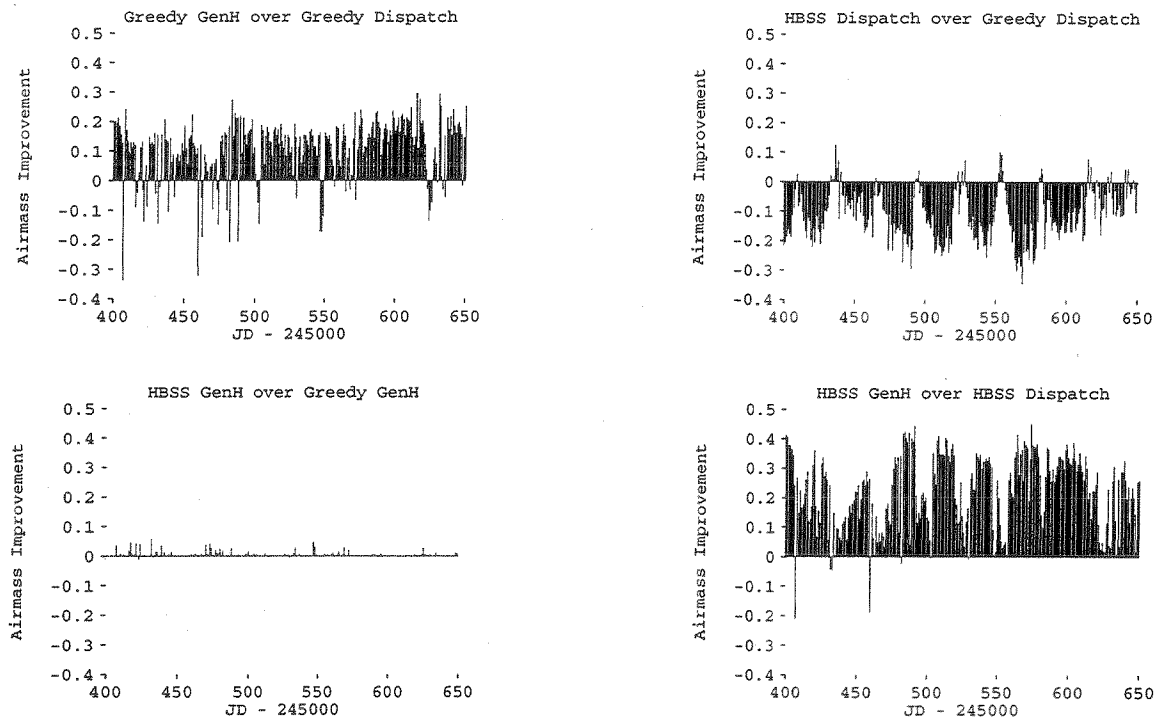

Figure 5: Improvement with respect to the airmass attribute score; HBSS performance is based on the best score after 15 samples averaged over 5 runs.

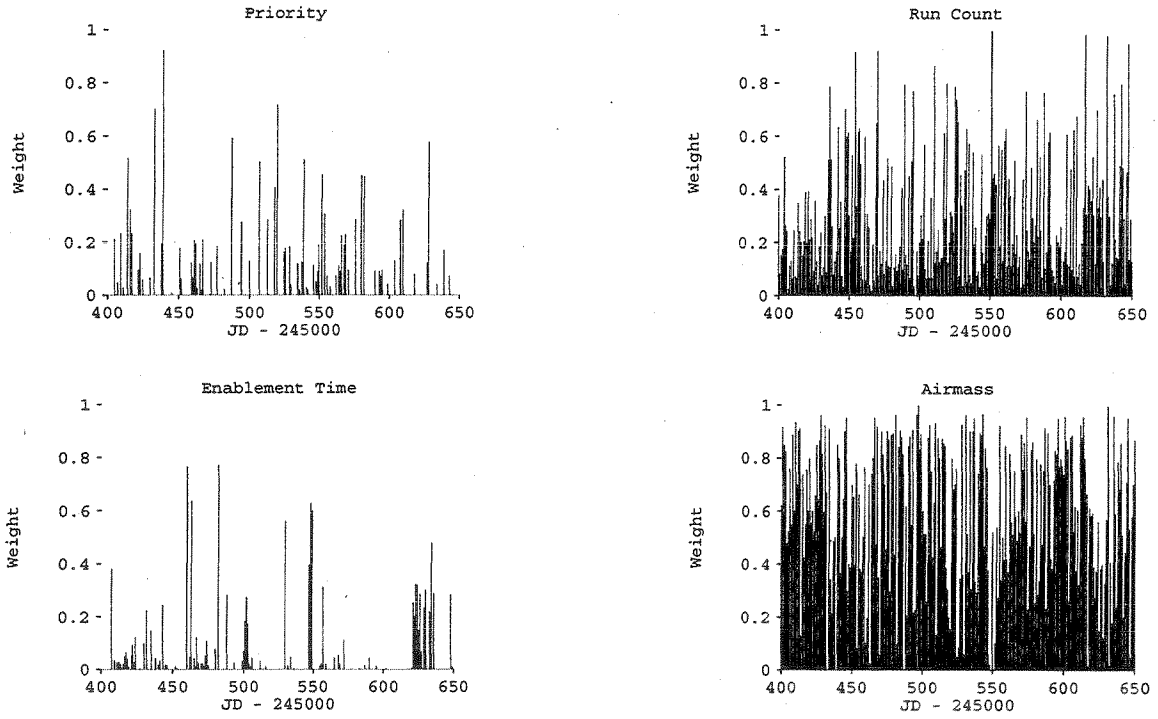

Figure 6: Weights generated by GENH for each of the four candidate attributes.