John Gipson: An Introduction to Sked, IVS 2010 General Meeting Proceedings, p.77–84
http://ivscc.gsfc.nasa.gov/publications/gm2010/gipson2.pdf

An Introduction to Sked

John Gipson

NVI, Inc., Code 698, NASA Goddard Space Flight Center, Greenbelt, MD, 20771
e-mail: `john.m.gipson@nasa.gov`

Abstract

In this note I give an overview of the VLBI scheduling software *sked*. I describe some of the algorithms used in automatic scheduling and some *sked* commands which have been introduced at users' requests. I also give a "cookbook" for generating some schedules.

1. Introduction

In this note I give an introduction to the VLBI scheduling software *sked*[1] widely used to schedule geodetic VLBI sessions. Nancy Vandenberg began writing *sked* as a graduate student in 1978. *Sked* has grown and evolved over time, becoming ever more sophisticated. Some highlights are summarized in Table 1. Vandenberg maintained *sked* until I took over this function in 2002.

Sked reads and writes *.skd* files which are special ASCII files that contain a complete description of the session, the schedule, and additional information used in scheduling the session. To schedule the session, *sked* needs to model many aspects of the VLBI system. The *.skd* file contains sufficient information to do so including:

1. Stations: Position, antenna slew rates and accelerations, horizon masks.
2. Equipment: Rack, recorder, recording options.
3. Source: Position, flux model.
4. Frequency setup: Number of channels, frequencies, bandwidths, 1 or 2 bit recording.

Sked can be run in either manual or automatic mode, or as a mixture of the two. Manual mode gives the user complete control over the schedule: for each scan they can specify the source, stations, start-time, and duration. Manual mode was used to schedule all sessions during the first decade of VLBI. However, as the number of stations and sources in sessions increased, it became increasingly cumbersome to schedule them manually. Currently a typical R1 session has 7 stations, 60 sources, and 1000 scans. Assuming it takes the scheduler only one minute to schedule a scan (which is actually quite optimistic) it would take almost two eight-hour days to schedule an R1.

Automatic scheduling was introduced in the early 1990s. The initial algorithm used covariance optimization—the user specified a set of parameters they were interested in estimating, and *sked* built up the schedule, scan-by-scan. At each step it would generate a set of trial scans and would choose the scan that, after being added to the schedule, would reduce the RMS sum of the formal errors of the parameters being optimized by the greatest amount. Although this approach seems reasonable, it led to schedules that looked strange to human eyes, and hence this approach

[1] *Sked* should not be confused with the similar sounding *sched* used to schedule VLBI astronomy sessions and maintained by Craig Walker of NRAO.

Table 1. *Sked* History Highlights.

1978	Basic program created by Nancy Vandenberg. Command line input. Manual selection of scans. Catalogs for sources, stations, equipment.
1981	Automatic calculation of antenna motion and tape handling.
1988	Automatic selection of observations (Autosked). Optimization by strict covariance.
1992	Evaluation of schedules using SOLVE simulations. Creation of pseudo-databases to evaluate formal errors.
1993	Autosked merged into standard version. “Strange” schedules caused autosked not to be used.
1995	Beginning of rule-based schedules.
1996	Mark IV/VLBA recording mode support added. Last time *sked* documentation updated.
1997	Support for VEX files. Y2K fixes. New Java-based catalog interface. (Too slow, so generally not used.) S2 and K4 support.
2002	John Gipson takes over development/maintenance. Fill-in mode. Best-N Source Selection.
2004	Linux port by Alexey Melnikov. Beginning of death of HP-*sked*. Astrometric option: Specify min, max observing targets for set of sources.
2005	Full support of Disk-based recording: Mark 5A, Mark 5B, K5.
2006	Downtime: Ability to specify when an antenna is unavailable.
2007	Resurrection of covariance optimization. Found and fixed various bugs in algorithms. Still not used routinely. By-product: *sked* can predict formal errors internally.
2008	Master command: Read session setup from master file. Check session against master file.
2009	Station limit raised from 32 to 64, and made parameter.
2010	Update documentation. Begin process of removing obsolete code for handling tapes.
2011	Make VEX native format.

was ultimately abandoned. Subsequently Vandenberg developed an alternative rule-based system based on a set of criteria about what constitutes good and bad schedules. For example, generally speaking, you do not want to observe the same source twice in a row. This leads to a rule saying ‘do not observe a source if it has been observed in the last X minutes,’ where X is user settable.

Quite often, different criteria of goodness are in conflict. On the one hand you want schedules that have good sky distribution—that is, that sample as much of the sky as possible in a short period of time. This can lead to schedules where the antennas spend a lot of time slewing from one source to another. On the other hand, you want to maximize the number of observations, which means you want to reduce slewing time.

In the next section we give an overview of the general algorithm currently used in automatic scheduling. This is followed by a review of a few useful *sked* commands. Lastly we give a cookbook example of scheduling a typical R1 session.

2. Overview of Automatic Scheduling

In automatic mode the scheduler instructs *sked* to generate a schedule until some end time, and new scans are added to the schedule, scan-by-scan, until the finish of the last scan is past the end time. This process is illustrated schematically in Figure 1 and described in more detail below.

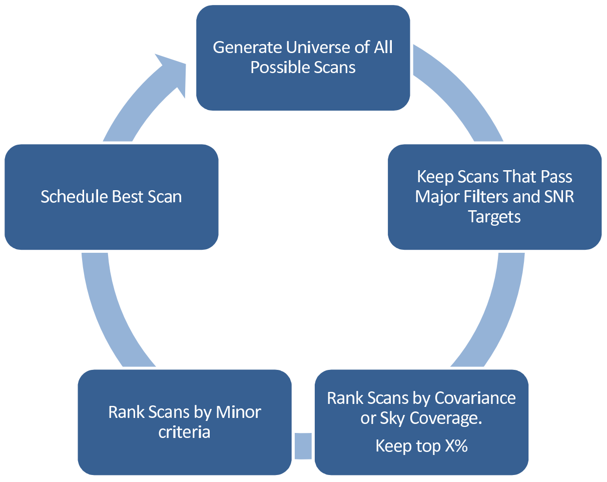

Generate Universe of All Possible Scans
Keep Scans That Pass Major Filters and SNR Targets
Rank Scans by Covariance or Sky Coverage. Keep top X%
Rank Scans by Minor criteria
Schedule Best Scan

Figure 1. Schematic illustration of automatic scheduling.

1. *Sked* calculates the source visibility table (a table which indicates which stations can see each source). This is used to generate the universe of possible scans.
2. The *major options*, together with SNR targets, determine which of the scans are actually generated and kept for further consideration.
3. *Sked* does an initial ranking by either covariance or sky coverage. The best $X\%$ are kept.
4. The *minor options* are used to assign a score to each scan based on several critera.
5. The scan with the highest score is scheduled.
6. This process is repeated until there is no more time.

2.1. Major Options

Major options determine which scans are generated and kept for later consideration. Many of the *major options* (top of Table 2) describe properties a scan must (or must not) have to be kept for further consideration. In this sense they are hurdles, because a scan is rejected if it fails to pass any of them. The remaining *major options* (bottom of Table 2) determine scan generation in other ways. For example, the third stage in automatic mode is to perform an initial ranking of the scans and keep the top *X%*. How this ranking is done, and the percentage kept, is controlled by *SkyCov* and *Best*.

Table 2. Major Options.

Hurdles	
Option	Brief Description.
AllBlGood	Yes: All baselines must meet SNR targets. No: At least 1/2 of the baselines for each station must meet SNR targets.
MinAngle	Minimum angular distance (degrees) between scans.
MinBetween	Minimum time (minutes) between observations of the same source.
MinSunDist	Minimum angular distance of a source from the sun.
MaxSlewTime	Maximum allowable slew time of an antenna to source
MinSubNetSize	Smallest Subnet to schedule.
Other Major Options	
SkyCov	Yes: Do initial ranking by sky coverage. No: Do initial ranking by covariance.
Best	% of scans kept for further consideration.
NumSubNet	Number of subnets to schedule simultaneously.
FillIn	Yes: Try to schedule additional observations using 'idle' antennas. No: Wait until all antennas become idle before scheduling.
FillMinTime	Minimum time (seconds) an antenna must be idle before it can be used.
FillMinSub	Minimum subnet size to schedule in FillIn mode.
FillBest	% of scans to keep for further consideration.

2.2. Minor Options

Minor options are a set of 13 criteria used to evaluate scans. Unlike the *major options*, where a scan must satisfy all of the criteria or it is rejected, a scan can be selected if it does well on some but not all of the criteria. Each of the *minor options* corresponds to a possible positive aspect of a scan. For example, all things being equal, a scan with more observations is better than one with fewer. The *minor option* that instructs *sked* to prefer scans with more observations is *NumObs*. The scheduler can select which options to use and how much weight to assign to them. A typical schedule will use 3–6 options. For each scan, *sked* calculates a score for each of the options in effect. The total score for the scan is the weighted sum of the sub-scores, and the scan with the highest total score is scheduled.

3. Some Useful *Sked* Commands

Sked currently has 76 commands, and in this brief note it is impossible to touch on more than a few. Many of the more recent commands originate from user feedback, and in this section I will

Table 3. Minor Options.

Option	Preferentially select scans with/that:
Astro	Astrometric sources that don't meet targets.
BegScan	Begin soon after previous scan ends.
EndScan	End soon after previous scan ends.
LowDec	With low dec sources.
NumLoEl	Low elevation sources.
NumRiseSet	Sources rising or setting.
NumObs	More observations.
SkyCov	Better sky coverage.
SrcEvn	More even distribution of observations/source.
SrcWt	Involving certain sources.
StatEven	More even distribution of observations/stations.
StatWt	Involving certain stations.
StatIdle	Sources that are idle.
TimeVar	Minimize variance of end times by station.

briefly describe three of these.

3.1. Master Command

Although it is possible to design a schedule 'from scratch'—that is, starting from nothing—most schedules are based on previous schedules. Figure 2 is a schematic illustration of how this works, and the steps involved in this process are summarized below.

1. Start with a similar schedule file and copy it, e.g., `cp r1412.skd r1414.skd`.
2. Open the file using *sked*.
3. Delete all observations.
4. Change the session code inside the schedule file.
5. Change the start and stop times of the session. This is usually done using information from the master file.
6. Change the stations. This is usually done using information from the master file.
7. If necessary, select a new frequency sequence.
8. Set the SNR targets.
9. Select the sources.
10. Generate the schedule.
11. Review and modify schedule if neccessary.

Many of these steps are prone to error, the most frequent being wrong start and stop times or wrong stations. These can prevent the schedule from being run at all. To make scheduling easier, I introduced the *master* command. This command has two modes:

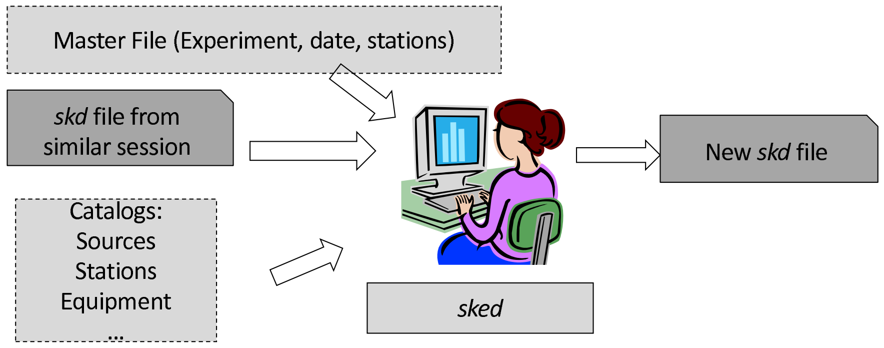

Master File (Experiment, date, stations)
skd file from similar session
Catalogs: Sources Stations Equipment ...
sked
New *skd* file

Figure 2. Writing a new schedule.

1. *Master check* compares the schedule against the master file, using the session code contained in the schedule, and issues a warning message if it finds any discrepancies in the station list or the start and stop times.
2. *Master get* initializes the schedule file using information from the master file. The session code in the schedule file serves as a key. *Sked* sets the start and stop times and the station list. It deletes all observations. If the SNRs of the original session were uniform by band, *sked* sets the SNRs. If not, it issues a warning message. The frequency sequence is left untouched.

At NASA Goddard we always use *master get* to initialize schedules. I encourage other centers to do so, not only because it makes it easier for the scheduler, but also because it reduces the possibility of error. Before being posted on the data center, all schedules are checked for consistency with the master schedule using *master check*.

3.2. *Downtime* Command

Frequently a station is unavailable for some part of the session. The most common reason is that it is participating in an Intensive session. Previously this situation was handled by the scheduler who would do the following:

1. Generate a schedule using all stations until some station(s) became unavailable.
2. Change the subnet to remove these station(s) from the network.
3. Continue generating until these station(s) again become available.
4. Change the subnet to include the stations that were unavailable.
5. Continue generating until the end of the schedule or another station becomes unavailable.

None of these steps is particularly difficult. However, they are all prone to error. To simplify the process of generating a schedule when some stations are unavailable for part of it, I introduced the *downtime* command. Using the *downtime* command the scheduler specifies an interval when a station or subnet is unavailable ('down'), and *sked* will automatically ignore these stations during this interval.

Sked employs the following algorithm in *downtime*. After *sked* generates a preliminary scan it checks to see if a station is down during the scan. If so, it rejects it from further consideration. A station is down (see Figure 3) when:

1. The end of the scan occurs while the station is down.
2. The begining of the scan occurs while the station is down.
3. The station is down during the scan.

Originally only the first two tests were done. These were sufficient if the station was down for a long interval, for example, during an Intensive. In January of 2010 a scheduler tried to use a *downtime* of less than 3 minutes, and an unanticipated situation occured. *Sked* generated a trial scan which started before *downtime* began, and ended after *downtime* ended, and hence it passed the first two tests. The user did not know why *downtime* failed—only that it failed. When the cause of failure was investigated and found, the *downtime* algorithm was modified to include criterion 3.

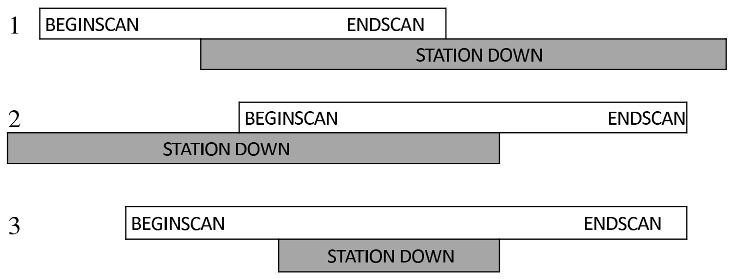
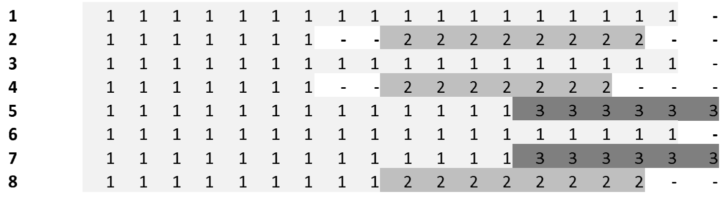


Figure 3. Scans which are invalid because of *downtime*.

3.3. Fill-in Mode

By default *sked* waits until all stations are free before scheduling a new scan. This can lead to periods of time in the schedule when many stations are idle, waiting for the rest of the stations to become free. I introduced Fill-In mode to reduce the amount of idle time.



1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-
2	1	1	1	1	1	1	1	-	-	2	2	2	2	2	2	2	2	-	-
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-
4	1	1	1	1	1	1	1	-	-	2	2	2	2	2	2	2	-	-	-
5	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-
7	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3
8	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	-	-

Figure 4. Using Fill-In Mode to reduce idle time.

Recall that which scans are considered for scheduling is governed by the major options. *Sked* will only attempt to schedule scans during idle time if *Fillin* is set to *Yes* and there are at least *FillMinSub* stations which are idle for at least *FillMinTime*. If successful, it repeats the process until either the amount of idle time or the number of idle stations drops below their threshold values. A typical result is displayed in Figure 4. The original scan involves all of the stations. Following this, *sked* schedules an additional scan using stations 2, 4, and 8, and then one more scan using stations 5 and 7.

4. Cookbook Example for Generating a Schedule

In this section we present a recipe for generating an R1 schedule (see Table 4). This same procedure, with obvious changes, can be adapted to generating other schedules. For most routine sessions the entire time required to generate a schedule is around ten minutes or less. Prior to the introduction of many of the labor-saving commands in *sked* it could easily take hours or days to generate a simple schedule.

Table 4. Scheduling an R1 Schedule.

#	Command line	Brief description
1	*linux_prompt*> cp r1412.skd r1413.skd	Copy an old schedule.
2	*linux_prompt*> sked r1413.skd	Open schedule in *sked*. *Sked* displays lots of information about the schedule.
3	? param exper r1413	Change internal session code. Required to use master command.
4	? master get	Initialize schedule based on master file.
5	? SNR ␣ X 20	Set X band SNR to 20. May not be required.
6	? SNR ␣ S 15	Set S band to 15. May not be required.
7	? Best 60	Get the best 60 sources for current network.
8	? Auto ␣ end	Generate the schedule until the end time.
9	? summ ␣ ␣ ␣ li	Do a summary listing of the schedule to review.
10	? Wr	Write the schedule to disk.
11	? Quit	Quit.

5. Conclusions

Sked continues to grow and mature with VLBI. Some of the changes in *sked* are driven by changes in equipment. Many changes are driven by user requests to make *sked* easier to use and less error prone. At Goddard we use *sked* to generate all of our VLBI schedules. I welcome all suggestions from users on improving *sked*. I give especially high priority to fixing bugs.