USE OF EVOLUTIONARY ALGORITHMS FOR TELESCOPE SCHEDULING

-

Ruud Grim[1], Mischa Jansen[1,2], Arno Baan[1,2], Jano van Hemert[2] and Hans de Wolf[1]
[1] Fokker Space, Newtonweg 1, Leiden, The Netherlands
[2] Leiden Institute of Advanced Computer Science - Leiden University,
Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands

r.grim@fokkerspace.nl
Tel. +31-71-5245416 FAX +31-71-5245835

Abstract

LOFAR, a new radio telescope, will be designed to observe with up to 8 independent beams, thus allowing several simultaneous observations. Scheduling of multiple observations parallel in time, each having their own constraints, requires a more intelligent and flexible scheduling function then operated before.

In support of the LOFAR radio telescope project, and in co-operation with Leiden University, Fokker Space has started a study to investigate the suitability of the use of evolutionary algorithms applied to complex scheduling problems. After a positive familiarisation phase, we now examine the potential use of evolutionary algorithms via a demonstration project. Results of the familiarisation phase, and the first results of the demonstration project are presented in this paper.

1. Introduction

The Fokker Space company has been active in space telescopes (e.g. ANS, IRAS, ISO, SAX) since 1968. Recent participation in telescope projects including VLT Delay Lines and LOFAR, both earth based telescopes, in which Fokker Space can re-use its gained engineering capabilities and technical expertise from space projects.

The role of Fokker Space in the current phase of the LOFAR project is providing system engineering support (including Reliability, Availability, Maintainability, Safety) and the preliminary development of the LOFAR Specification and Scheduling Segment.

Existing telescope schedule engines do not provide algorithms that deal with the intrinsic complexity of the LOFAR concept in relation to constraints imposed by simultaneous observations and complex resource sharing.

Within the internal Fokker Space R&D programme, a series of studies have started to investigate the suitability of the use of evolutionary algorithms applied to complex scheduling problems. Here we report on the results of the familiarisation phase that was recently finalised.

2. The scheduling problem in LOFAR

2.1 Highlights of LOFAR

LOFAR's goal is to open up a new, high resolution window on the lowest region of the electromagnetic spectrum (10-220 MHz) that is accessible from the earth [1]. The design highlights currently envisaged offer future users an instrument that is much more flexible than traditional radio telescopes have been [2]. The main characteristics are:

- Independently steerable multiple electronic beams.
- The ability to rapidly point to any position in the accessible sky.
- To process the high data rate on a processor dedicated to such tasks (and possibly outside the capability of conventional processing techniques and generally available hardware).

Some relevant background information on LOFAR is provided in the next sections. Relevant documentation can be found at www.lofar.org.

2.2 Science with LOFAR

Currently, the following key science areas are expected to be supported [1]:

1. **The High Redshift Universe:** Low frequency radio observations have proven to be an efficient means of detecting high red shift radio galaxies, which in turn can be used to probe the onset of structure formation in the Universe. Because these sources are rare, detecting them will necessitate large-scale surveys of the entire sky accessible to LOFAR.

2. **The Epoch of Reionization**: The collapse of the first structures in the Universe should have produced 21 cm radiation from the surrounding neutral hydrogen. Today, highly red shifted in LOFAR's observing range, this emission, if it can be detected, can be used to map the first structures to form in the Universe.

3. **Mapping the 3-Dimensional Galactic Cosmic Ray Distribution:** At low frequencies H II regions become opaque. If the H II region is at a known distance, the intensity of non-thermal emission in front of it can be compared with that along nearby lines of sight to infer the synchrotron emissivity toward the H II region. LOFAR has the potential to apply this method to hundreds to thousands of H II regions in the Galaxy. If successful, the result would be a 3-D map of the cosmic-ray emission in the Galaxy.

4. **Bursting and Transient Phenomena:** the flexibility and reconfigurability of LOFAR offers great potential for the study of variable phenomena. The unknown nature of many of these effects require maximum flexibility of the instrument, to ensure it can be used efficiently.

5. **Active and Passive Solar and Ionospheric Applications:** Astronomical and terrestrial sources of radiation not under LOFAR control can be exploited to probe the ionosphere or the environment of the Sun on short time scales and fine spatial scales. Examples include interplanetary scintillation of radio sources and passive radar from FM broadcasts. The development of a separate transmit facility is beyond the current LOFAR project. If an outside group(s) were to develop one, it could be combined with LOFAR to conduct bi-static radar mapping of the solar corona.

2.3 Observing with LOFAR

LOFAR is designed to measure in a frequency range of approximately 10-250 MHz. The frequency range 10-80 MHz is called the low frequency range, and the range 100-250 MHz is called the high frequency range.

Observations required to take place in the high frequency range need to be scheduled at night because of ionospheric conditions, while the low-frequency obervations can occur at any time. It is also not very likely that low-frequency and high-frequency observations will be performed at the same time.

2.4 Acquisition stations

Physically, the telescope consists of more than 10.000 antennas spread over an area with a diameter of more than 300 kilometres.

Figure-1: An artist impression of a possible LOFAR configuration in the Netherlands (courtesy ASTRON)

Antennas are grouped together as substations, and several substations form one station. About 25% of all antenna's will be placed in a virtual core of about 2 km diameter. All other antennas are placed within stations (for an impression see Figure-1) along one of the spiral arms.

Figure-2: An artist impression of a LOFAR antenna station in the Netherlands (courtesy ASTRON)

2.5 Beamforming

Beamforming is the process where the telescope is configured to “look” into a certain direction (phased array). In contrast to more traditional telescopes, where the location of the object to measure follows immediately from the orientation of the dish, LOFAR accomplishes this via hardware and software. The underlying idea is that signals from celestial objects take different times to reach the different antennas.

LOFAR allows up to eight beams to be used simultaneously because of three facts:

- Incoming signals are now processed with the help of embedded software instead of analogue electronics.
- Beamforming is done by the digital processing part.
- A very high-capacity network & processing capacity will be installed that allows massive amounts of data to be transported and processed.

The multi-beam aspect is perhaps the most remarkable feature of the telescope. However, all antennas are needed in the process of beamforming, meaning that if some part of the system fails, e.g., a station, this will affect all beams. This will certainly influence measurements, but does not necessarily yield worthless data.

2.6 Scheduling

Job scheduling is a difficult task, and this also holds for telescope scheduling. In general, scheduling problems are NP-hard (NP = non-deterministic polynomial, meaning that there are no known algorithms that can give a solution to the problem in polynomial time [3].

Obviously, the main goal of telescope scheduling is to maximise observing efficiency by maximising the number of high quality observations, minimising time it takes to execute observations, minimising wasted observations, and optimising the use of a limited resource, the telescope itself. On the other hand, we would also like the scheduler to be fair, so non-urgent, but important observations will not be put off all the time in favour of more urgent ones.

This is complicated due to the complex resource sharing in the LOFAR instrument. Some resources (like antennas) can be shared by simultaneous observations, but other resources (such as processing pipelines) cannot be shared and observations will compete for their availability.

The software design of LOFAR uses abstractions called observation types (related to the kind of observation) and virtual instruments (an abstraction to capture the necessary resources for specific purposes). These abstractions will hide instrument details from the scientists and will enable flexible operation of the telescope and future easy-to-be-implemented new observation types. Furthermore, as mentioned before, in principle LOFAR is able to execute up to 8 parallel observations using its 8 virtual beams.

After some investigation we concluded that current telescope scheduling systems, such as SPIKE [4] or SWAS [5], do not include the functionality and flexibility that is expected to be required to solve the (parallel) scheduling problem of LOFAR and therefore, at a very early stage of the LOFAR project, Fokker Space, and supported by the Leiden Institute for Advanced Computer Science, decided to investigate the possible use of Evolutionary Algorithms for telescope scheduling within its internal R&D programme.

3. Evolutionary Algorithms

3.1 Introduction

Nearly all practical scheduling problems can be described in terms of the job-shop scheduling problem [6]. This problem can be described as follows. Consider a manufacturing environment in which n jobs need to be done on m machines. Jobs typically require processing on several machines and for every job there is a set of constraints on the order in which the machines can be used as well as a processing time on each machine. Now the aim is to find the sequence of jobs on each machine in order to minimise a given objective function, usually being the total ‘makespan’, i.e., the time to finish all jobs.

Solving a job-shop scheduling problem with an evolutionary algorithm is done by applying specialised operators that allow for meaningful crossover and mutation of candidate schedules. A schedule in this case is a specification of the use of all machines, indicating when to do what part of which job on this machine. Because of the complex nature of the scheduling problem there is almost always a need to use heuristics at some level. Finding these heuristics is in itself a difficult problem, and is studied among others in the field of Operations Research.

3.2 Principles of Evolutionary Algorithms

Evolutionary Algorithms (EA) are a modern implementation of Darwin's principle of the-survival-of-the-fittest. EA's are computational models inspired by evolution. These algorithms encode potential solutions to a specific problem in a chromosome-like data structure. Each of these chromosomes represents an individual. Decoding and evaluating the chromosome of an individual indicates how good this potential solution actually is. This is known as the fitness of an individual. The group of individuals is called the population.

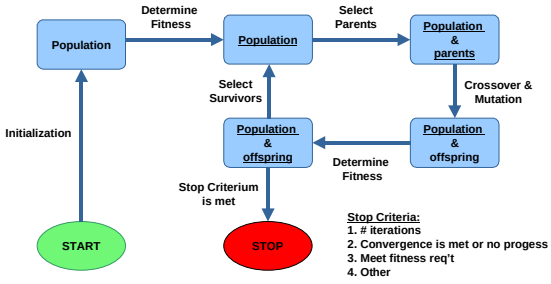

Figure-3: A visualisation of an evolutionary algorithm

At the start of the algorithm a population is set up. The evolutionary process then decides which members of the population are most likely to reproduce, based on the fitness of the solution they contain. The individuals that represent a better solution to the target problem are more likely to reproduce than those that represent a poor solution.

After reproduction a new population enters the same process. This process is repeated using cross-over, mutation and selection, until certain given constraints are satisfied or until a given amount of populations have been examined. Cross-over and mutation principles are further discussed in Section 4.6.

Whether a problem can be solved using an evolutionary algorithm, depends on a few basic factors. It must be possible to determine the fitness of a solution, and to determine which solution is close to an acceptable solution. Evolutionary algorithms assume that combining two good individuals will probably create even better offspring.

4. A first prototype of the EA schedule engine

4.1 Introduction

After studying the theory and principles of EA's we decided to build an early and simplified prototype to demonstrate the EA principles and to familiarise with them.

Some of the details of the EAs for scheduling mentioned here are implemented. The concept of hard constraints and soft constraints or preferences is used in the implementation of the scheduling engine. Most others can be implemented if there is a need for it.

First, we have defined a simple concept for observation database. The scientist defines a macro observation, the schedule operator will work with micro observations. In some cases they may equal.

4.2 Schedule model

In our simplified telescope schedule model we specified a macro observation using the following entry values:

- Macro Id and Observation Type
- Cpu-use
- Start-stop dates & times
- Length, Co-ordinates, Frequency
- Hard constraints (e.g. during night or day; after another observation)

Macro observations will be split into one or more micro observations which will have a fixed duration in time.

The micro observation is specified using:

- Micro Id & Macro Id (Parent)
- Length & Co-ordinates
- After (i.e. relation to other observations)

4.3 Schedule model constraints

The following constraints were implemented in our schedule model:

- The object must be visible (a simplified model is used)
- Resource may not exceed maximum value
- All observations are at same frequency or simultaneous observations
- Needs to be scheduled after another observation
- Not allowed to follow other observation immediately
- Needs to be a strict observation (e.g. fixed in time)
- Needs to be done during day/night

One can think of many other (realistic) constraints that should or could be implemented in the final schedule engine. However, for simplicity, we disregard all of these here.

4.4 Definition of the Evolutionary Algorithm

In defining an Evolutionary Algorithm we specified the following ingredients (Figure 3):

- Chromosome representation (chromosome & decoder)
- Initialisation
- Fitness function
- Genetic operators (crossover, mutation)
- Selection (parent, survivor)
- Termination criteria

The chromosome is the series of micro observations. A micro observation is a gene. The decoder takes up a micro observation and will put it on the chromosome. Initialisation will create a pool of individuals necessary to start the evolutionary process.

First strict observations (e.g. fixed in time and/or frequency) will be placed. E.g. high frequency observations will be allocated first in the blocks reserved for high frequency. Then repeating observations (e.g. from a survey) will be placed. Next the other micro observations will be placed up front as possible. In this phase constraint satisfaction is not strived for; this will be achieved by applying the fitness function.

This fitness function will be applied to all individuals, or rather to the schedule that is created from them by the decoder. For every individual two fitness values will be computed; the so-called *constraint-fitness* and the *preference-fitness*. Individuals scoring zero points on the constraint-fitness component represent valid schedules, and those that also score *high* on the preference-fitness component are the better individuals among those that are valid.

Note that a high constraint-fitness value indicates a non-fit individual. Those having low constraint-fitness are the better solutions, although still incomplete ones and probably not totally valid, and a constraint-fitness value of zero indicates a complete and valid solution.

Note that in our model we use a positive number to indicate the badness of an individual.

4.5 Fitness Function

4.5.1 Constraints

If the constraint fitness of an individual, that represents a possible schedule, is not equal to zero, it is an invalid solution. However, it can be repaired by removing some of the observations. This (interactive) functionality is not yet implemented in our model.

Implemented constraints in the model are:

- exceeds available amount of resources
- not all observations at same frequency
- execution order violated
- follows earlier observation too soon
- failed to schedule some observations

The (non-)violation of these constraints can be represented as a vector; the so-called constraint break (cbreak) vectors which will be used in the calculations.

4.5.2 Preferences

Implemented preferences in the simulator are:

- high above the horizon

This was done using a very simplified model as illustrated in the Figure 4 and 5.

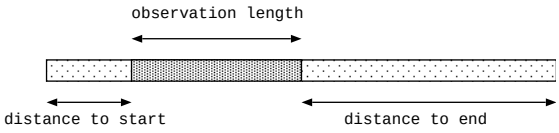

Figure-4: Placing an observation in a specific scheduling window

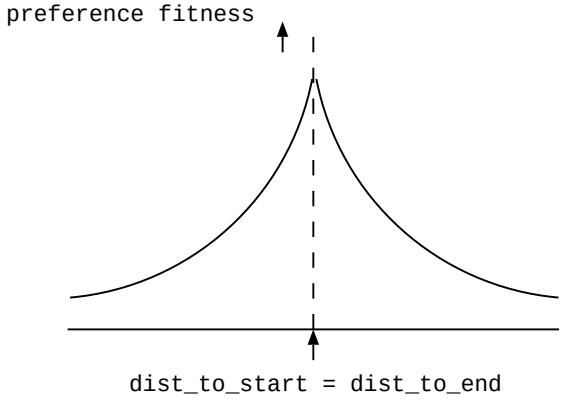

Figure-5: Preference value

4.5.3 SAW vector

The SAW-vector is introduced to determine the weights, used for calculating the constraint-fitness, during the run. SAW stands for “Stepwise Adaptation of Weights” and is described in [7].

Since it is not known how severe each constraint violation should be punished, and which constraint will be the most violated and which the least, choosing weights for punishment is very hard to do. Therefore we choose to make the SAW-vector part of the algorithm. The SAW-vector indicates the weight of punishment per constraint type. This method avoids having us experiment a lot with different weight settings for the constraint. What happens is that during the run the weights are updated and these should give a fair impression of the relative impact of every constraint. This information can be used for later runs.

4.5.4 Fitness computation

The actual computation of the values of the 2 fitness-components is as follows:

- *constraint fitness*

  Multiply for each individual the SAW-vector with the individual's cbreak-vector. Note that an individual that has few constraint violations has a low constraint fitness.

  CF = (c1s1 + c2s2 + c3s3 + c4s4 + c5s5)

- *preference fitness*

  For all observations, except for micro survey observations, measure the preference fitness value. The sum of all these numbers is the preference fitness for this individual.

  PF = sum [obs_length/(dist_to_start – dist_to_end)+1]

4.6 Operators

In creation of new individuals several operators are being used. The principles are sketched below.

4.6.1 Crossover

The crossover (Figure 6) step combines the chromosome of the two parents into a new child. When this is done to create a schedule special attention must be given to this operator to ensure that the child chromosome represents a valid permutation of the observations in the schedule.

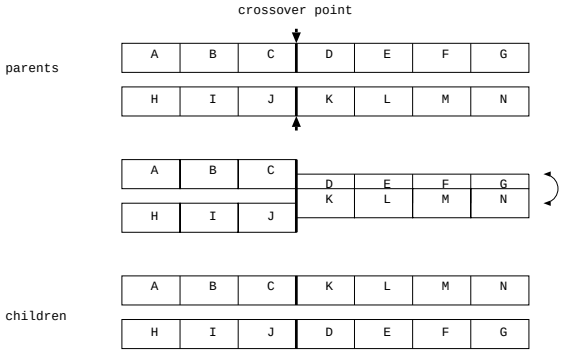

Figure-6: Crossover

4.6.2 Mutation

Mutations introduce some random variations in the chromosomes. Also for mutations (see Figure 7) special attention is required to keep the chromosome a valid permutation.

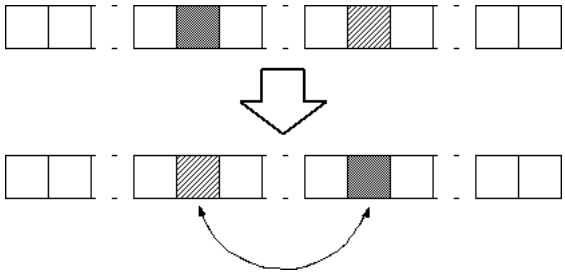

Figure-7: Mutation

4.7 Selection

The selection mechanisms are used to select parents or to select survivors. Parent selection is used to select individuals from the current population that will act as parents that can breed and create offspring. Survivor selection is used to select the offspring that is allowed to live on, and possibly gets to replace some individual or individuals in the current population. This also keeps the population size within strict bounds.

A steady state approach, replacing only a few individuals at a re-iteration, is adopted because it has efficiency benefits over the generational model which replaces the whole population at every iteration [8]. Furthermore, we used a population of size 30.

4.7.1 Parent selection

We use linear ranking selection for parent selection, and we select two individuals to act as parents per selection step. Linear ranking selection is a good method to prevent a very fit individual from taking over the entire population

4.7.2 Survivor selection

We use a tournament selection method among the two offspring and the two worst individuals in the parent pool to select the two individuals that make it to the parent pool. These four individuals are compared against each other, and the two most fit make it to the parent pool.

4.8 Termination criterion

A decision has to be made as to when the algorithm should stop. The aim of any EA is to find a good solution in reasonable time, and if possible to find the optimal solution. Since it is not known what the optimal solution is, we will have to define when a solution is acceptable.

First of all, it seems reasonable to limit the run time of the algorithm. If no good solution is found, it may be that the problem should be relaxed somewhat. The runtime of the algorithm is related to the number of fitness evaluations performed, so we defined an upper limit of 20.000 iterations for this.

4.9 Repairing individuals

If the termination criterion is reached, it might happen that there is no valid solution, i.e., one that has a constraint-fitness value equal to zero. In that case, we can create a valid schedule by removing observations until all constraint violations are resolved. Strictly speaking this also does not yield a valid solution, because not placing all observations is also considered to be a constraint violation. However, individuals that violate this constraint can still represent schedules that are suited for execution.

In LOFAR it is foreseen that the scheduling system not only contains an automatic scheduling tool, but also an interactive scheduling tool, allowing the scheduler to repair or overwrite automatically generated schedules.

5. Testing

5.1 Test definition

The goal of evaluation is to investigate how well the EA works, and how the settings of the different parameters affect the quality of the schedules produced by the algorithm. Since there are a lot of parameters, this gives us a lot of combinations to explore. It is unfeasible to explore all of them, or even a large part of them. Therefor, in total 19 test cases were defined to evaluate the EA of the schedule simulator. In these, also invalid test cases are incorporated in order to investigate how the EA deals with them.

5.2 Test runs

After the test cases are created, we start the EA with the defined settings. For every test case provided we run the algorithm 10 times, with different values for the random seed, allowing us to compute average performance afterwards. We need to have a different random seed for each run since we are using a pseudo-random number generator.

5.3 Test results and evaluation

It goes to far to describe here in detail the results of all the tests executed. As a matter of fact, the simulator used in the familiarisation study and all the results have been described in great detail in the master's thesis of Mischa Jansen [9].

At this stage we want to summarise the major findings of the EA research.

- *Proper design of the encoding of a observation schedule request in a chromosome and decoding the chromosome into a observation schedule is essential*

A natural way to represent a schedule in a chromosome is to define a gene for every time slot. This approach leads to an unwanted break up of observations that spend time slots. An indirect representation was used, where the sequence in the chromosome determines the order in which the decoder places the observations into a schedule.

This decoder requires careful design. It is tempting to make this decoder intelligent (using knowledge of constraints and preferences when placing the observations), but this may interfere with the operation of the EA. In such situation certain permutations are ruled out which could lead to a better result.

- *None of the runs resulted in a completely valid schedule*

The results indicate that special measures are needed to come up with valid schedules, since the "best" individuals still violate some of the constraints. Therefore the schedules cannot be used directly, but rather some post processing is required to turn them into valid schedules. Some thoughts on this are given in the recommendations.

- *Evolutionary computation seems to be a promising approach*

The goal of the specific EA study was to examine if evolutionary computation techniques can be used to solve the LOFAR observation scheduling problem. Based on the results we can conclude that the EA performs good enough, since the algorithm shows convergence towards a good solution. Although the results are not always valid, the evolutionary approach certainly promises to be a good approach to handling the problem. For a more successful scheduling engine additions and improvements to the EA are required.

- *Comparison of the results against other methods is difficult, since we only implemented an EA to solve the problem*

Since we only implemented an EA we cannot compare the results to other more traditional techniques. Furthermore, the results obtained are all based on test cases created by the test case generator. We cannot be sure that these test cases compare to problems as will be encountered when the telescope starts functioning.

5.4 Recommendations

Based on the preliminary test results the following recommendations were made:

- Continue with the EA
- Improve the decoding-step of the algorithm
- Repair invalid schedules, or at least remove observations until a valid schedule emerges
- Experiment with overbooking the schedule. This requires the test case generator to be adapted, since it cannot create overbooked schedules
- Allow the scheduler to repair and override the automatic scheduling
- Define a good test & validation plan for the development phase

6. Conclusion

The first results of the study demonstrate that evolutionary algorithms indeed seem suitable for implementation in a telescope observation schedule engine. The next step will be improving the schedule simulator with LOFAR representative functionality, such as observation types and virtual instruments. Together with other functionality's that will be developed by Fokker Space within its R&D program in support of LOFAR, the schedule simulator will be implemented as the schedule engine within a prototype for the LOFAR Specification & Scheduling Segment. The first results of improving the schedule engine are expected by Summer 2002. Incorporation of this engine in the prototype is expected in the second half of 2002.

7. Acknowledgements

We acknowledge ASTRON, one of the initiators and co-ordinators of the LOFAR telescope development, for providing valuable information during interviews and the use of photographic material from the LOFAR project.

8. References

[1] LOFAR Science User Requirements Document; see www.lofar.org

[2] LOFAR Technical Documentation; see www.lofar.org

[3] Computers and Intractability: A Guide to the Theory of NP-Completeness, Michael R. Garey and David S. Johnson, 1979, W.H. Freeman & Co..

[4] Portable Astronomical Scheduling Tools, Glenn E. Miller and Ashim Bose, 1996, Astronomical Data Analysis Software and Systems, 5

[5] Small, Fast and Reusable: A Satellite Planning and Scheduling System, Steven Kleiner, 1998, Astronomical Data Analysis Software and Systems, 7.

[6] Genetic Algorithms for Scheduling, Philip Husbands, 1994, AISB Quarterly, No. 89

[7] Graph Coloring with Adaptive Evolutionary Algorithms, A.E. Eiben, J.K. van der Hauw, and J.I. van Hemert, 25--46, 1998, Journal of Heuristics, 4(1).

[8] Genetic Programming, An Introduction, Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone, 1998, Morgan Kaufman Publishers, Inc. and dpunkt-Verlag für digitale Technologie GmbH.

[9] A Scheduling Simulator for LOFAR, Master's Thesis, Mischa Jansen, Leiden University, Nov. 2001