A scheduler for the National Aures Observatory

Y A Bekhti1, K Lamraoui2, N Seghouani1, K Benatchba2, S Bouzidi-Hassini2

1CRAAG, Route de l'Observatoire, BP 63, Bouzareah, Algiers, Algeria
2ESI, BP 68 M, Oued Smar, 16270, Algiers, Algeria

nseghouani@yahoo.com

Abstract. The main project for National Aures Observatory (NAO) is the construction of a Meta-Telescope consisting of several robotic and ultra-rapid telescopes. This telescope ‘farm’ will face hundreds of observation requests that have to be scheduled. The goal of this work is to compute the best schedule for all observations, over all night and using all available telescopes. The scheduler must also take into account all the interruptions that may randomly occur during the observation process, especially those relative to Gravitational Waves and Gamma-Ray Bursts alerts, which are the main goal of the Aures Observatory. To solve this multi-objectives optimization problem, we proposed a scheduler of a predictive class. A genetic algorithm approach was used to compute the solution of the scheduling problem based on NSGA-II and using Pareto Optimality.

1. Introduction

Scheduling problems are known to be a hard task in the problem optimization field, in fact they are classified as an NP-Hard problem. Therefore, implementing a solution for this kind of problems is equally difficult and requires the use of advanced and sophisticated methods and algorithms in order to come up with satisfying results. The scheduling of astronomical observations is no different in terms of difficulty and complexity. Indeed, the development of new instruments that use new technologies has transformed the field of astronomical observation to a new era and a wide range of possibilities are now available to astronomers. For us, the most important advent would be ultra-rapid robotic and autonomous telescopes which will be implemented in the upcoming National Aures Observatory (NAO) in Algeria during the following years.

Autonomous observatories like the NAO, are nowadays able to execute a high number of observations per night that cover different types of targets, from simple observations of stars and planets to more complex ones like variable stars, tracking GRBs, asteroid occultation and observing Electro-Magnetic (EM) counterparts of gravitational waves (GW) events. This high-level environment has made inevitable the use of scheduling technics in order to satisfy the observatory objectives and contribute to its success.

Other observation schedulers exist and are operational, with most of them being used for only one telescope. Some others are used to compute the observation program for a network of independent telescopes, but rare are those implemented for a set of telescopes working together for a common goal, which is our case. This means that our scheduler will have to compute a program for all telescopes together, not independently. This will make possible the scheduling of complex observations, like...
simultaneous GRB tracking using different filters and a rapid search of EM counterparts for GW events.

We will discuss some of the existing schedulers in section 2 of this paper and we will approach the NAO scheduler in section 3. In the latter, we will explain and define the problem elements in 3.1, which are mainly the constraints and objectives of the scheduler. Then, we will have more details on how to deal with multi-objectivity in 3.2 and finally how are we going to compute the schedule in 3.3.

2. Existing Schedulers

Many observation schedulers have been implemented in the past, either partially or fully autonomous but most of them rely on merit functions, which is used to calculate a target benefit for an observing program (http://www.rts2.org/rts2-flwo.pdf). A merit function is constructed according to the needs and objectives of each observatory and thus are different from one observatory to another, but each one of them has to take into consideration the altitude of the observed target, the best altitude being its transit point which is the best observing condition for any target (without taking into consideration weather conditions). Other possible parameters that can be added to the merit function are the observation priority, number of targets to be observed, the use of resources (telescopes and filters) and other position related parameters such as distance to the Moon. In this section, we briefly describe three studied schedulers, Majordome I, Majordome II and RTS2.

The Majordome is Tarot's first managing system, with a scheduling unit as one of its modules. It was first implemented in 2000 [1] for the scheduling of GRBs observations and other common observations for the Calern Observatory. Users with different priorities send their observation requests to the Majordome in order to be scheduled by the scheduling unit. The scheduling process works by scheduling observations one by one, starting by the ones with higher priority, then observations with less priority. Each observation is inserted in its best observing time considering the target's altitude. This is a direct calculation method which may not be optimal if we need to fulfill several objectives [1].

Majordome II is another managing system for the Arago telescope in Paris Observatory. With most of the basic concepts of the first Majordome kept, this new version introduced another way of scheduling observations. The scheduling unit does not calculate just one schedule for the night, but rather many possible schedules. In this scheduler, the night-sky is divided into several areas and a schedule is calculated for each one of them independently, thus constituting what it is called a plan scheme. The execution unit chooses, for each area, observations with the highest priority first, then those with less priority, and so on. This plan scheme is also used to deal with possible disruptions like GRBs alerts, inadequate weather conditions or resource failures. The execution unit has then the possibility to change the area from the plan scheme so as to choose one that fits the current situation after the end of the disruption [2].

Another scheduler which makes used of meta-heuristics, unlike previous direct methods, is the scheduling unit of the Remote Telescope System 2 (RTS2). Its scheduler offers three scheduling modes: dispatched scheduling, queue scheduling and planned scheduling. In the first mode, targets are chosen and observed one at a time, which means that there is not any complete schedule for the whole night. A merit function value is calculated for all targets and is used to select the best observation to execute first. After the observation is done, the values of the merit function are recalculated for the rest of observations and a new one is chosen. Queue scheduling is one of the most used scheduling technic which utilizes a built ordered list of targets to be observed, called a queue (http://www.rts2.org/rts2-flwo.pdf). It is partially automated as human intervention can be introduced to either build the queue, change the order of observations or even change the entire queue. The execution unit then observe each target following the defined order. The planned scheduling is simply a user built schedule to be executed by the execution unit, it is thus a manual mode. In its newest version [3], RTS2 uses a meta-heuristic for scheduling the night due to the high number of objectives that needs to be optimized in the process. This multi-objectivity aspect is dealt with using an adequate genetic algorithm, the Non-
Dominated Sorting Genetic Algorithm II (NSGA-II). This is the same method used in our work, although the problem parameters differ. We will discuss more about multi-objectivity and NSGA-II in section 3.2 and 3.3 respectively.

3. The NAO scheduler

The NAO observatory is an undergoing project consisting of more than 10 ultra-rapid and robotic 50cm telescopes. All telescopes will be used to fulfill a common goal. This means that scheduling the set of observation requests will have to be done for all telescopes in one go and not for each one of them independently, adding one more parameter to the problem, the number of telescope used. The telescopes will be used for common observations, GRB tracking, search for EM counterparts of GW events, asteroid occultation, stellar variability, and numerous types of observations, all put into service for the national and international scientific community. This adds to the importance of the scheduling unit and its role in such a complex environment.

Naturally, scheduling observations has to be done under a set of constraints, most of them being observational, and has to optimize the observatory objectives for better scientific results. We will break down these elements and try to describe each one of them in the following sub-section. Additionally, we will also discuss the method used for solving this scheduling problem.

3.1. The problem’s elements

Constraints and objectives constitute the main elements of a scheduling problem. Here we focus on those of the scheduling of the NAO observation requests. The schedule to be calculated has to satisfy, in the most complete way possible, all constraints set, and it also has to be the most optimal possible.

3.1.1. The Constraints. Our constraints can be divided into two groups: user constraints and observational constraints. In the first group, we have the priority of the observation request owner, which will later define the priority of the observation, and also the user observation quota. Users with exhausted quota will have its priority changed to the lowest level, which is reserved for this purpose. Observational constraints are either explicitly defined by the user or implicitly set by the system. This set of constraints is as follows:

- The upper and lower bounds for the observing time: the user can specify the requested time interval when his observation must be launched. If not defined, the scheduler will choose the appropriate time based on observing conditions (altitude, distance to the moon) and the user priority.
- The observation frequency: defines how many times the observation should be repeated. the interval between each observation is either defined by the user using the period parameter or left for the scheduler. In the latter case, each observation will be scheduled independently.
- The observation period: indicates the period between multiple observations of the same request. Only when the observation frequency is set.
- Exposure time.
- Rise, set and transit times: these are calculated values that are used to define when is the target visible and when is its maximal altitude, which is its best observing condition, independently on the weather.
- Minimal altitude: gives the target's altitude allowed for the observation to take place. If not set, default value is used based on the observatory dome and observing conditions at small altitudes.
- Minimal distance to the Moon: the distance allowed between the observation target and the Moon. If not set by the user, the default value is used.

3.1.2. The Objectives. In order to evaluate the schedule quality, we need to fix the observatory goals which will define our objective function. However, due to the complexity of the problem, we need more than one objective function to express all goals and thus evaluate the schedule properly. It is
natural that one of the objectives is to maximize the scheduled observations so as to satisfy as many requests as possible. Another important aspect is the quality of observation. Thus, targets have to be observed in their best position possible. Finally, since we have many operational telescopes, we aim at using the least telescopes possible for all observations. Again, here are the scheduler objectives:

- **Scheduled observation**: the number of scheduled observations needs to be maximal.
- **Observation altitude**: targets need to be observed near their transit, meaning that the altitude should be maximal. The closest position to transit needs to be chosen for all targets.
- **Number of used telescopes**: in order to have telescopes ready for unexpected events, like GRBs, we need to use the least telescopes possible to schedule all observation requests.

All these objectives are translated into mathematical objective functions that will be used to evaluate the calculated schedule and decide whether or not it is acceptable.

### 3.2. Multi-objectivity with Pareto Optimality

Dealing with multi-objectivity can be done with several methods, like the weighted single objective function or keeping one single objective and moving the others to the set of constraints. In the first method, each objective function is assigned a weight that represent its importance among other objectives. A new and single objective function is then constructed by adding all the objective functions multiplied by their respective weight. In the second, one chooses a single objective function and moves the others to the constraints set. However, these two methods have major inconveniences when it comes to decide which objectives are more important than other, by either fixing a more important weight for them or choose which one to keep. This is a very difficult problem to solve and requires for example machine learning technics for fixing the weights, which is another and independent optimization problem.

Luckily, the Pareto Optimality overcomes the inconveniences of previous cited methods by finding an equilibrium between all three objectives without any use of weights [4]. The schedule evaluation is done using the Pareto dominance principle, a solution (a schedule) is then said to be Pareto optimal if it isn't dominated by any other solution.

**Definition:** A solution \( \mathbf{X} \) dominates another solution \( \mathbf{Y} \) if and only if

\[
\forall i \ (x_i \leq y_i) \land \exists j \ (x_j < y_j)
\]

Where \( x_k \) and \( y_k \) are values of the \( k \)-th objective function for \( X \) and \( Y \) respectively.

From this definition, we deduce that many possible schedules can be Pareto optimal and not only one. This set of Pareto Optimal solutions is called Pareto optimal front. This particularity guided us for choosing the appropriate method for calculating the schedule using Pareto Optimality.

### 3.3. Computing the schedule

Having a set of possible schedules as Pareto Optimal solutions make us think of population based evolutionary algorithms as the best fit for searching such set of solutions. Indeed, two main algorithms for finding the Pareto optimal front can be found in the literature: the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [5] and the Strength Pareto Evolutionary Algorithm (SPEA2) [6], both are good for finding the Pareto optimal front. However, NSGA-II offers more diversity regarding the front solutions, unlike SPEA2 which offers fast convergence instead [5]. We choose NSGA-II because we are more concerned about having different kind of solutions in the final set. Also, the convergence time of NSGA-II is acceptable.

#### 3.3.1. NSGA-II

It is a genetic algorithm [7] suitable for finding Pareto optimal front. It adds other methods and operators to basic genetic algorithm operators which are the selection, crossover and mutation. In the following we give a short description by [5]:

1. **Selection:** The selection process is crucial and affects the performance of the algorithm. It determines which solutions are chosen to be parents for the next generation.
2. **Crossover:** Crossover is a process where two parent solutions are combined to produce offspring solutions.
3. **Mutation:** Mutation introduces diversity into the population by randomly altering some of the genes.

The NSGA-II algorithm consists of several steps:

1. **Initialization:** A population of candidate solutions is randomly generated.
2. **Evaluation:** Each solution in the population is evaluated based on the objective functions.
3. **Selection:** The selection process is performed, choosing the best solutions to proceed to the next generation.
4. **Crossover and Mutation:** Genetic operators are applied to create new solutions.
5. **Ranking:** Solutions are ranked based on their dominance relationships.
6. **Crowding Distance:** Solutions are further ranked by their crowding distance, which measures the density of solutions in the neighborhood.
7. **Archiving:** The best solutions are kept in an archive, which is used to ensure diversity in the population.
8. **Replacement:** The new population replaces the old one.
9. **Termination:** The algorithm continues until a termination criterion is met, such as a maximum number of generations.
The step-by-step procedure shows that NSGA-II algorithm is simple and straightforward. First, a combined population $R_t = P_t \cup Q_t$ is formed. The population $R_t$ is of size $2N$. Then, the population $R_t$ is sorted according to non-domination. Since all previous and current population members are included in $R_t$, elitism is ensured. Now, solutions belonging to the best non-dominated set $F_1$ are of best solutions in the combined population and must be emphasised more than any other solution in the combined population. If the size of $F_1$ is smaller then $N$, we definitely choose all members of the set $F_1$ for the new population $P_t + 1$. The remaining members of the population $P_t + 1$ are chosen from subsequent non-dominated fronts in the order of their ranking. Thus, solutions from the set $F_2$ are chosen next, followed by solutions from the set $F_3$, and so on. This procedure is continued until no more sets can be accommodated. Say that the set $F_i$ is the last non-dominated set beyond which no other set can be accommodated. In general, the count of solutions in all sets from $F_1$ to $F_i$ would be larger than the population size. To choose exactly $N$ population members, we sort the solutions of the last front $F_i$ using the crowded-comparison operator $\leq_n$ in descending order and choose the best solutions needed to fill all population slots. The NSGA-II procedure is also shown in Fig. 6.1. The new population $P_t + 1$ of size $N$ is now used for selection, crossover, and mutation to create a new population $Q_t + 1$ of size $N$. It is important to note that we use a binary tournament selection operator but the selection criterion is now based on the crowded-comparison operator $\leq_n$. Since this operator requires both the rank and crowded distance of each solution in the population, we calculate these quantities while forming the population $P_t + 1$, as shown in the above algorithm.

In figure 1 we give the components of the NSGA-II algorithm.

![Diagram showing the main loop of NSGA-II](image)

3.3.2. Chromosome representation. A chromosome in genetic algorithm is a solution to the optimization problem, which in our case is the observation schedule. Our chromosome must then have all information regarding the schedule. Each chromosome is a set of Genes. A gene is the basic modelling unit which contains information about a single observation. The number of genes in our case is then equal to the number of observation requests to be scheduled. In the following we give the components of a gene:
• **Start date of the observation**: indicated the exact date and time of the scheduled observation.

• **Duration**: represents the total duration of the observation, which is the exposure time added to the slewing and preparation time of the telescope and processing time of the image.

• **The allocated telescope**: gives the ID of the telescope to be used for this observation.

### 3.3.3. Genetic Operators.
Crossover and mutation are applied each with a given probability. The crossover is applied on two parent schedules and consists of:

1. Pick a random index between 1 and the number of genes.
2. Construct the first resulting schedule using observations of the first parent schedule.
3. Add the rest of observations from the second parent schedule.
4. Repair the resulting schedule as it can be not feasible (in case of an overlapping observations).

The mutation operator is applied on a single schedule and it consists of changing randomly one of the values of the gene. The resulting schedule is also repaired if not feasible.

### 3.3.4. The selection.
The selection method is offered by NSGA-II. It is a crowded binary tournament dominated selection. Every individual in the population has the non-domination rank and the crowding distance as one of its attributes. We add another attribute which is the number of violated constraints. The selection is then as follows:

\[
i <_{n} j \begin{cases} 
 j_{\text{violation}} > 0 \land i_{\text{violation}} < j_{\text{violation}} \\
 \text{or} \ (i_{\text{rank}} < j_{\text{rank}}) \\
 \text{or} \ (i_{\text{rank}} = j_{\text{rank}} \land i_{\text{distance}} > j_{\text{distance}})
\end{cases}
\]

### 3.4. First implementation
During our first attempt to implement this scheduler, we programmed a random observation generator in order to get observations to be scheduled. We generated approximately 300 observation per test. Our first try implementing the NSGA-II algorithm resulted in a computational time nearing the 20 seconds with the mentioned number of observations. During this process, we adjusted the genetic algorithm parameters (crossover and mutation probabilities).

Currently, we are implementing a newer version of the scheduler by taking into account code optimization, using real data from Tarot observatory and comparing different types of methods. Results will be published in the near future.

### 4. Conclusion
During this first step of implementing the observation scheduler, we have been able to clearly identify and define the scheduling problem we are facing and its parameters (constraints and objectives). The problem objective is composed of multiple objectives and require an adequate method for more interesting results, we chose Pareto Optimality in our case.

Our first try implementing NSGA-II to solve our scheduling problem resulted in a 20 seconds’ execution time. During the next steps, we will be trying to reduce this number by optimizing the code, trying different methods and also involving parallelization processes like Open-MPI.

### References
[1] Bringer M et al 2001 *Experimental Astronomy* 12 33
[2] Moisan S et al 2002 *Proceedings of SPIE : Observatory Operations to Optimize Scientific Return III*, 4844, 262
[3] Kubánec P 2010 *Advances in Astronomy* **2010** id902484
[4] Kubánec P 2010 Master Thesis *arXiv* 1002.0108
[5] Deb K et al 2002 IEEE Transactions on Evolutionary Computation **6** 183
[6] Zitzler E et al 2001 TIK-Report **103**
[7] Mitchell M An introduction to genetic algorithms. 1998 *MIT press*