Test benchmark instances for the Orienteering Problem with Synchronization

Jorge Riera-Ledesma
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
jriera@ull.es

April 30, 2019

Abstract

The Orienteering Problem with Synchronization is an optimization problem arising in the management of a telescope observation instrument. This instrument consists of a set of configurable devices that allow the simultaneous study of numerous astronomic objects, so the observation of an object may require the synchronized configuration of several devices. Astronomers using this instrument propose a list of objects to observe. Because the time window assigned to each proposal is quite limited, the astronomers associate a priority with each object.

The Orienteering Problem with Synchronization aims at selecting the objects to observe. The problem also includes determining an optimal sequence of configurations for each device to maximize the total priority of the selection, subject to some synchronization issues and a time limit to conclude the last observation and set the devices in a final configuration.

This paper describes a set of two-hundred and forty benchmark instances designed to compare the performance of algorithms on different hardness instances.

1 Introduction

Gran Telescopio Canarias (GTC)[1] is a research project in Astronomy which administrates the use of a modern 10.4 m telescope also called GTC and located in the island of La Palma (Canary Islands). The Regional Government of the Canary Islands, the Spanish Government, European Funds, and also non-European partners (*Instituto de Astronomía de la Universidad Nacional Autónoma de México* and the *University of Florida*, among others) actively support the GTC. A defining aspect of this scientific infrastructure is its ability to deliver cutting-edge data through new instruments.

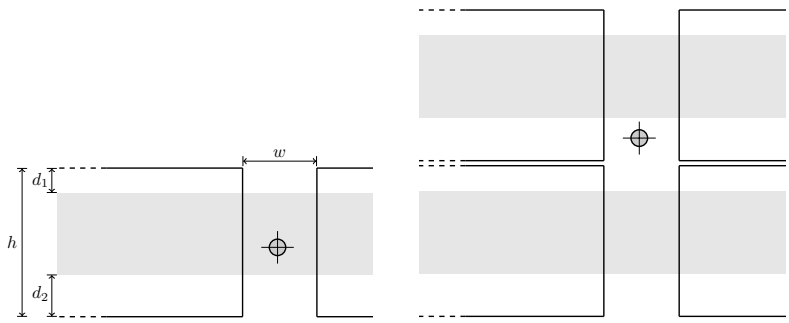
[1] http://www.gtc.iac.es/

One of the most recent instruments in GTC is Near Infrared Multi-Object Spectrograph (EMIR)[2], operational since 2017. It is a wide-field imager which is able to observe up to fifty-five slits over a $6.64' \times 4'$ spectroscopic field of view. Symbols $'$ and $''$ denote arcminute and arcsecond units, respectively.

EMIR divides its observation field into fifty-five contiguous and parallel bands. Objects in each band are observed by a device consisting of a couple of retractable opposite bars of height h''. Each device is configured to open a slit w'' surrounding each object to observe. The value h is fixed for the instrument. The value w depends on the brightness and size of the object, and makes EMIR capture its infrared spectrum.

Each band consists of three horizontal areas: an upper dead zone of size d_1'', a central area, and a lower dead zone of size d_2''. When an object is placed in the central area, it is observed by the device of the band. When an object is placed in the upper or the lower dead zone, it needs at least the devices of the nearest two bands. Furthermore, certain circumstances (e.g., the calibration of the instrument) make an object to require three o more devices. Summarizing, EMIR can perform many simultaneous observations, and some of the observations may require the synchronization of two or more devices.

Figure 1(a) shows the three areas dividing a band: its two opposite bars are setting up a slit to observe an object centered in between. Figure 1(b) shows two bands, each one subdivided into the mentioned three areas, and one object needing two devices.



(a) (b)

Figure 1: Details on the bands, slits, and dead zones

Every semester astronomers submit proposals of potential observations. Each proposal is characterized by a list of object locations, priority values, exposition times, and slit widths. On the base of this demand a panel of experts assigns time windows to each proposal. Then the astronomers have to restrict their initial lists to optimize the assigned resource. This optimization problem has been manually approached using a graphical user interface called EMIR Optimized

[2] http://www.iac.es/proyecto/emir/, http://www.gtc.iac.es/instruments/emir/

Slits Positioner (OSP)[3]. This tool helps the astronomers to assign objects to the fifty-five bands composing EMIR, and to determine which bars set up slits for each selected object, and which sequence they need to follow in order to perform the observations.

More precisely, given a proposal, the here-presented problem aims at selecting the objects to observe and the sequence of configurations performed by each device to make the selected observations. The objective is to process as many prioritized observations as possible within a stipulated time limit. Note that processing each observation might require the cooperation of multiple devices. We call this optimization problem the *Orienteering Problem with Synchronization (OPS)* as it shows characteristics of Orienteering Problem (OP) [1], well known in the context of Vehicle Routing Problems (VRPs).

We describe a set of two-hundred and forty benchmark instances designed to compare the performance of algorithms on different hardness instances. Section 2 gives a precise definition of the OPS, and Section 3 details the generation features of this test bed set.

2 Problem setting

The OPS may be considered as a job scheduling problem where each device represents a configurable processor, and each potential observation represents a job, whose processing might require the synchronized configuration of several processors. Then, OPS looks for a selection of jobs and a sequence of configurations for each processor, to perform as many prioritized jobs as possible, within a time limitation. Performing a set of jobs consumes setting times of the implied processors, waiting times associated with the synchronization, and processing times for the jobs. We now describe the input parameters, characterize a feasible solution, and illustrate with a small example.

2.1 Input parameters

We denote by $J = \{1, \ldots, n\}$ the set of potential jobs. No job in J is compulsorily required to be performed.Each observation associated with a job $j \in J$ requires a processing time of p_j units of time. Processing a job j produces a non-negative revenue b_j, associated with its priority of observation. Setting up a processor k from the configuration to process job i to the configuration to process job j consumes a setup time d_{ij}. For convenience of notation, let $t_{ij} = p_i + d_{ij}$. We also define for modeling purposes jobs 0 and $n+1$, that are an initial and a final dummy jobs, respectively. Both configurations represent a safety status parking each device in the middle of the view area. Let $p_j = b_j = 0$ for $j \in \{0, n+1\}$, and $V = J \cup \{0, n+1\}$. The observation instrument behaves like a set $K = \{1, \ldots, m\}$ of configurable non-identical parallel processors, so that each job j requires a subset $K_j \subseteq K$ of processors to be performed. All

[3] http://www.iac.es/proyecto/emir/pages/observing-with-emir/observing-tools/osp.php

processors in K_j need to set up a configuration at the same time to perform j during at least p_j time units. Similarly, we denote by $J_k \subseteq J$ the set of jobs that require the processor k to be performed. Let $V_k = J_k \cup \{0, n+1\}$.

Let $G_k = (V_k, A_k)$ be the *transition graph* for the processor k, where A_k is the set of arcs among all vertices in V_k, except the ones leaving $n+1$ or entering 0. Let $G = (V, A)$ be the aggregated graph where A contains all the arcs. Note that G is a multi-graph since two vertices may be linked by several arcs. Hence A is a set of triplets (i, j, k) such that $(i, j) \in A_k$ and $k \in K$, and each triplet $a = (i, j, k)$ is associated with a length t_a equals to the time t_{ij}. Finally, let L be the given time limit to perform the selected jobs. We assume that $L \geq t_{0j} + t_{j,n+1}$ for all $j \in J_k$ and $k \in K$.

2.2 Feasible solutions

A feasible solution for the OPS is characterized by one elementary path $P_k \subset A_k$ from 0 to $n+1$ in G_k for each $k \in K$. Each path represents a sequence of jobs starting and finishing at the parking positions. Not all jobs need to be visited by a path, but when a path visit a job j then all paths p_k with $k \in K_j$ must visit j. In addition there must exist a value s_j for each $j \in V$ satisfying:

$$s_i + t_{ij} \leq s_j \qquad (i, j) \in P_k, k \in K \tag{1}$$

$$s_{n+1} - s_0 \leq L. \tag{2}$$

The value s_j represents the starting time to perform a job j, and together with constraints (1) guarantee the synchronization of the processors when performing each job. That is, all processors involved in the processing of a specific job may reach the configuration to start processing the job at different times, but they all need to start the process at the same time. Inequality (2) limits the overall processing time.

The aim of the OPS is to find the paths P_k and the values s_j maximizing the sum of the profits associated with the selected jobs.

2.3 An example

This section presents input parameters and a feasible solution for a small OPS instance. Figure 2 represents an input consisting only of three devices focusing on five potential observations. Objects 1, 4 and 5, located in middle areas, require exactly one device to be observed. Objects 2 and 3 fall in between dead areas, so they need to be observed by their surrounding devices: devices 1 and 2 are required to observe the objects 2, while the object 3 requires devices 2 and 3 to be observed.

More precisely, in this instance $J = \{1, 2, 3, 4, 5\}$, $K = \{1, 2, 3\}$, $J_1 = \{1, 2\}$, $J_2 = \{2, 3, 5\}$, $J_3 = \{3, 4\}$. The processing times are given by $p = [0, 1, 1, 1, 1, 1, 0]$, and the profit values are given by $b = [0, 1, 1, 1, 1, 1, 0]$. The transition cost c_{ij} are given by the horizontal distance between the objects. The time limit is $L = 11$. The graphs in Figure 3 depict valid transitions for each processor.

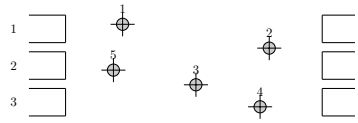
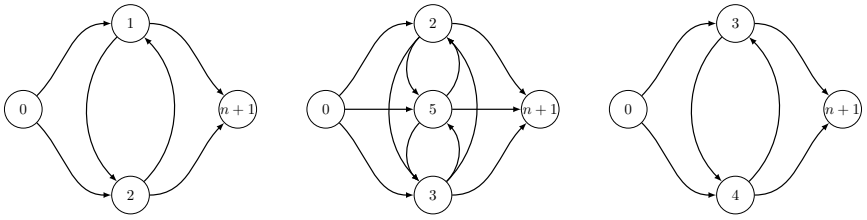


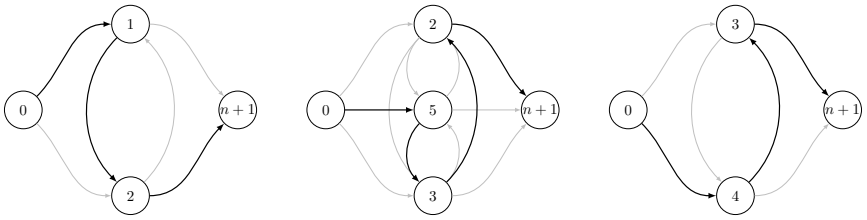
Figure 2: Example with five jobs and three processors



(a) G_1 (b) G_2 (c) G_3

Figure 3: Transition graphs for the example in Figure 2

We now present a feasible solution for this instance. This solution starts setting up the processors 1, 2, and 3 to perform jobs 1, 5, and 4, respectively. Then, processors 2 and 3 are set up to process job 3. Finally, processor 1 and 2 are configured to perform job 2. Note that we are assuming all processors start from configuration 0 and finish at configuration $n + 1$. Figure 4 shows the elementary paths on the transition graphs.



(a) P_1 (b) P_2 (c) P_3

Figure 4: Elementary paths for the example in Figure 2

Valid scheduling times values, according to (1) and (2), are given by $s = [0, 1, 8, 6, 4, 2, 11]$. The diagram shown in Figure 5 helps to understand the validity of this solution. We can observe in this diagram idle times of processors due to the need of synchronization.

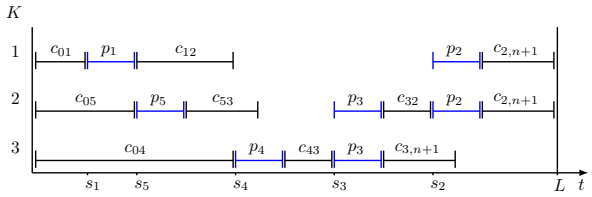


Figure 5: Diagram associated with a feasible solution for the example in Figure 2

3 Test instances

We have built a battery of artificial benchmark instances based on the description of the instrument EMIR given in Section 1. We define five sets of sources of random potential targets. Each set of target locations consists of five hundred items. Each item i from each set is characterized as a triplet $((x_i, y_i), p_i, b_i)$, where (x_i, y_i) are coordinates within the $240.0'' \times 398.4''$ square defined by the EMIR field of view, relative to an origin located at its center. Therefore, the horizontal coordinate value x_i is randomly selected in the continuous interval $[-120.0, 120.0]$, while the vertical value y_i is randomly selected in the continuous interval $[-199.2, 199.2]$. The processing time p_i is an integer number uniformly generated in the interval $[1, 200]$. The value of the prize b_i is randomly selected in the set $\{1, 10, 100\}$.

As pointed out above in Section 1, the EMIR field of view is horizontally divided by fifty-five bands of height $h'' = \frac{398.4''}{55}$. Each band is divided itself in three horizontal areas: an upper dead zone of size $d_1 = 1''$, a central area, and a lower dead zone of size $d_2 = 3''$. The *basic configuration* for observation requires that a target placed in the upper or the lower dead zone needs to be observed by slits set up by at least the nearest two sets of opposite bars, while if that target falls within the central zone it only needs just one band. The *calibration configurations*, however, require that each target is observed by the two, three, and four nearest bands. Thus, for our computational experiments to gather all these configurations, we have defined four classes of instances $\{A, B, C, D\}$. The class A represents the basic configuration and classes B, C, and D represent the calibration configurations involving two, three, or four bands, respectively.

We have ranged the size of the instances in such a way it allows us to evaluate the performance of the algorithms on easy and hard cases. A preliminary computational experience helped us to determine these limits: instances with smaller size became trivial, and instances with greater size consumed the computer resources. Thus, we have been ranged the number of observation targets n in the set $\{120, 130, 140, 150\}$ for Class A, Class B in the set $\{110, 120, 130, 140\}$, Class C in the set $\{60, 65, 70, 75\}$, and Class D in the set $\{45, 50, 55, 60\}$.

For a given family of sources, and a given class of instances, an instance of size n is generated by selecting the n first suitable targets from the chosen family of sources. By "suitable" we mean those targets able to be appropriately

processed by their assigned bands. So, any target falling within the upper dead zone of the first band, or within the lower dead zone in the fifty-fifth band is not considered for instances in Class A. Similarly, targets falling in the first and last band are not considered for classes B and C, and targets falling within the two first, and two last bands are not considered for class D.

Summarizing, we have generated $5 \times 4 \times 4$ target sets, from the five set of sources, four classes of instances, and four different sizes for each class. Each target set is stored in a file named `X_nY_Z.txt`, where `X` represents the class of instance, `nY` represents the size, and `Z` is an ordinal number.

Using each target location file we have generated a set of input instance files for the OPS. Each input instance is no longer a set of target locations, but a set of input parameters as described in Section 2.1. Given a file of target locations, an instance file contains, for each band $k \in K$, a sets J_k, priority prizes b_j, for each $j \in V$, a $V \times V$ matrix containing the values t_{ij}, for each pair of vertices $i, j \in V$, and a time limit L for the global processing. These ingredients allow the construction of the graphs G_k for each $k \in K$, to define the input instance ultimately.

The instance generator process generates the components t_{ij} for the cost matrix. We compute t_{ij} as $p_i \lfloor v d_{ij} \rfloor$, where p_i is the processing time for the job i, v is the bar configuration speed, and d_{ij} is the distance between to targets i and j. Aiming at a realistic design we have considered the horizontal movement of the sliding bar. The distance measure between two target points (x_i, y_i) and (x_j, y_j) is $d_{ij} = |x_i - x_j|$.

In the same fashion than Fischetti et al. [2] for the OP, we define the time limit for each instance as

$$L = \left\lceil \alpha \max_{k \in K} \mathrm{TSP}(G_k, t) \right\rceil,$$

where $\mathrm{TSP}(G, t)$ is the value of a minimum cost Hamiltonian tour in a set in a weighted graph G, with arc cost in t, and α is a given parameter. Note that for the case of instances using horizontal distance the value $\mathrm{TSP}(G, t)$ is trivially computed as $2|\max_{i \in J_0} x_i - \min\{0, \min_{i \in J_0} x_i\}|$. We consider three values for the parameter α, namely $\alpha = 0.25$, 0.50, and 0.75.

Summarizing this second stage, for each target file, and according to its respective class of instances, we generate three input instances for the OPS, taking into account three different values for L. Therefore, we have generated $5 \times 4 \times 4 \times 3$ input instances. Each instance is stored in a file named `X_nY_Z_`α`_W.txt`, where α represents one of the three possible values that this parameter might take, and `W` is an ordinal number.

4 Description of the files

This repository consists of three folders: `input`, `output`, and `generator`. The folder `input` contains four sub-folders associated with the classes A, B, C, and D, respectively. Each class folder consists itself of two folders containing

the target sets and instance sets, provided in *json* format. The folder `output` contains PDF files showing feasible solutions for each input instances. In some cases, these solutions are optimal solutions. The folder `generator` contains the source code for our target and instance generator. This code could shed light on details of the generation and shows the way for reading the input instances (see, e.g., files `EMIR_target_set_t.hpp` and `EMIR_instance_t.hpp`).

Acknowledgement

This research has been partially supported by the Ministerio de Economía y Competitividad research projects MTM2015-63680-R (MINECO/FEDER, UE) and ProID2017010132 (Gobierno de Canarias/FEDER, UE).

References

[1] P. Vansteenwegen, W. Souffriau, D. Oudheusden, The orienteering problem: A survey, Eur. J. Oper. Res. 209 (2011) 1–10.
URL https://doi.org/10.1016/j.ejor.2010.03.045

[2] M. Fischetti, J. J. Salazar-González, P. Toth, Solving the orienteering problem through branch-and-cut, INFORMS J. Comput. 10 (1998) 133–148.
URL https://doi.org/10.1287/ijoc.10.2.133

[3] J. Riera-Ledesma, Data set for the Orienteering Problem with Synchronization, Mendeley Data, Version 1 (2019).
URL http://dx.doi.org/10.17632/vmb6zb2827.1