Splitting Method for Spatio-temporal Sensors Deployment in Underwater Systems

Mathieu Chouchane[1], Sébastien Paris[1], François Le Gland[2], and Mustapha Ouladsine[1]

[1] LSIS, Aix-Marseille University, Domaine universitaire de Saint-Jérôme, Avenue Escadrille Normandie Niemen, 13397 Marseille Cedex 20, France
[2] INRIA Rennes, Campus de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes Cedex, France

Abstract. In this paper, we present a novel stochastic optimization algorithm based on the rare events simulation framework for sensors deployment in underwater systems. More precisely, we focus on finding the best spatio-temporal deployment of a set of sensors in order to maximize the detection probability of an intelligent and randomly moving target in an area under surveillance. Based on generalized splitting technique with a dedicated Gibbs sampler, our approach does not require any state-space discretization and rely on the evolutionary framework.

Key words: Evolutionary algorithm, Stochastic optimization, Generalized splitting, Genetic algorithm, Gibbs sampler.

1 Introduction

Consider a randomly moving target that tries to cross a closed area or to run away from a known position. We want to compute the best spatial and temporal deployment of sensors in order to maximize the detection probability of this intelligent and randomly moving target. Until recently, these types of problems were solved using operational research algorithms [8,12,14,19], which commonly model the constraints in both discrete time and space. In order conduct a continuous optimization, we have decided to use a novel probabilistic optimization algorithm based on the rare events simulation framework.

Probabilistic optimization algorithms are based on natural evolution processes of a population of individuals. Each of them represent a direction in the space of solutions to a problem. These algorithms have been developed by mimicking natural evolution and simplifying biological knowledge. All of them follow a simple scheme: (i) first, initialize arbitrarily a population and rate the fitness of each individual with a scoring function; (ii) using the individuals' fitness select a proportion of the population for reproduction. The selection process can be deterministic (for example select the best solutions) or completely random; (iii) then, generate a new population of solutions, from those selected previously, through a modification process that is often used to include recombination and/or mutation [1,13]. The new individual obviously shares many characteristics of its parents; (iv) we repeat the fitness-selection-reproduction process until certain stopping criteria are reached. Genetic algorithms have proven their efficiency for complex optimization problems. When the optimization is unconstrained or depends on "simple" or linear constraints, we can also use parametric methods such as covariance matrix adaptation [11] and its evolutions [6] or the cross entropy method [3,5] which is a parametric method based on the rare events simulation framework. However, when we are faced with a strongly constrained optimization problem, parametric optimization methods and other classic genetic algorithms cannot always handle the consistency of the solutions.

The goal of this paper is to present the method we have used to solve our real-world problem. This method is similar to the non-parametric genetic algorithms, but it is based on the generalized splitting method. This approach helped us to conduct a continuous optimization under a heterogeneous set of constraints. The splitting method mainly differs from a genetic algorithm in that a solution can remain in the pool for iterations before it is modified or excluded, while in most of evolutionary algorithms, the parents are ousted from the new generation. Moreover, in the reproduction step, we use a dedicated Gibbs sampler as a special mutation operator.

Along this paper, we will try to point out the commonalities and the differences between the two approaches. We will also try to justify why the splitting method may better suit these kind of problems.

Our article is organized as follows. The first section is devoted to the presentation of the real-world spatio-temporal scheduling problem. In the next two sections, we present our method in order to solve our optimization problem. We then apply and illustrate the splitting algorithm with the *flaming datum* [17,18] problem and offer some conclusions.

2 Problem Presentation: Spatio-temporal Search Efforts Planning

This section deals with the presentation of our real-world problem and its associated constraints.

Our goal is to maximize the detection probability of an intelligent and randomly moving target in an area Ω under surveillance. This can be achieved by optimizing the spatio-temporal deployment of a set of limited sensors during a period T. A spatio-temporal deployment consists of a list of sensors activations and a set of their associated position.

The target that we consider is not only smart but also reactive and its trajectory is unknown (and depends in practice on random variables).

2.1 The Solution Constraints

We have a set of P sensors $\boldsymbol{s}_i$ that we may spatially and temporally deploy in our operational theatre $\Omega \times [0, T]$ in order to detect a smart and reactive target. Let $\boldsymbol{X} \in \mathcal{X}$ a solution. So we can define

$$\boldsymbol{X} \triangleq \{\boldsymbol{s}_i(\boldsymbol{\tau}_i)\}_{i=1,\ldots,P} \text{ with } \boldsymbol{\tau}_i = \{t_{i,1}, \ldots, t_{i,j}\}_{j=1,\ldots,np_i}. \quad (1)$$

$\boldsymbol{s}_i$ corresponds to the sensor i ($i \in \{1 \dots P\}$) position, while $\boldsymbol{\tau}_i$ is a vector containing its np_i ($np_i \leq E_{max}$) instants of activation (in $[0, T]$). We also denote by $\mathcal{C}$ all the spatial and temporal constraints on the feasible solutions. As soon as a sensor is deployed (for example: dropped and set up), it is powered on and starts consuming energy; it becomes out of service when its battery is empty. Until that moment, it can send a limited number of pings (E_{max} times maximum).

Also, we assume that the sensors are only active and do not cooperate. In other words, they can only detect a target when they send a signal. Since they are not autonomous, they only ping when they are in the radio range of a moving commanding station that requests it. This constraint is denoted by the "operator" visibility parameter until time $t \leq T$, $\varphi_t(\boldsymbol{X})$.

2.2 The Target Constraints

A feasible trajectory of duration T is denoted by the random variable $\boldsymbol{Y}_T \in \mathcal{Y}$. We are only given an *a priori* on the starting point of the target trajectory. This *a priori* is weak if the starting point is randomly sampled in the search space Ω. On the contrary, a strong *a priori* means its initial position is sampled from a Gaussian pdf centred on the last seen position.

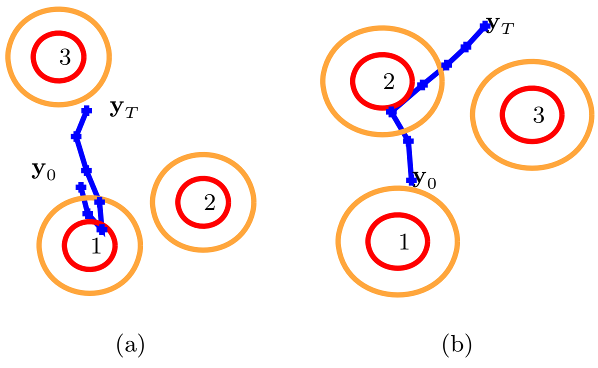

Fig. 1. (a) The target starts from $\boldsymbol{y}_0$. After a while, it is detected by an active sensor (sensor 1) and immediately escapes by following a radial course. (b) The target starts from $\boldsymbol{y}_0$. After a while, it detects an active sensor (sensor 2) and avoids it.

If the target detects a signal originating from a sensor, but is too far from it, it is able to avoid it before being detected while simultaneously memorizing its position. In our model, we define the target as detected if it enters the sensor's detection range (delimited by the red circle in figure 1). However, since the target is smart, if it comes close enough to a sensor which has just sent a signal (in the zone delimited by the red and orange circles in figure 1), it detects this sensor and learns all of its specifications. Thus, it may decide to avoid this threat or to come closer and start an avoidance later. In all cases, when the target is notified of the existence of a sensor, it changes its course before it enters the detection range. The target trajectory controls (for the avoidance of a sensor) are then directly influenced by the partial visibility (or knowledge) of the target on the deployed search efforts $\mu_t(\boldsymbol{X})$.

3 The Generalized Splitting Framework

The purpose of this section is to discuss the generalized splitting framework in the context of unconstrained (or simple constrained) optimization. This will be used in the next section.

Consider a random object $\boldsymbol{X}$ (vector, random variable, etc.) that takes values in some set $\mathcal{X}$ and is distributed according to a pdf f. Also consider a real-valued function S on $\mathcal{X}$, a threshold level $\gamma \in \mathbb{R}$ and assume that sampling from f is easy. Let $\gamma^\star = S(\boldsymbol{X}^\star)$. Most optimization problems involve finding $\boldsymbol{X}^\star$, defined by

$$\boldsymbol{X}^\star \triangleq \arg\max_{\boldsymbol{X} \in \mathcal{X}} \{S(\boldsymbol{X})\}. \tag{2}$$

Splitting theory is based on the observation that maximizing $S(\boldsymbol{X})$ is similar to estimating probabilities of the form

$$\ell(\gamma) = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \int_{\mathcal{X}} \mathbb{1}_{\{S(\boldsymbol{X}) \geq \gamma\}} \, f(\boldsymbol{X}) \, d\boldsymbol{X}, \tag{3}$$

given that this probability reaches zero when γ converges toward the optimal score $S(\boldsymbol{X}^\star)$. Maximizing $S(\boldsymbol{X})$ is also similar to sampling the set

$$\mathcal{X}_\gamma = \{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) \geq \gamma\} \subset \mathcal{X}. \tag{4}$$

with the idea that this set decreases toward $\boldsymbol{X}^\star$ when γ increases toward the unknown value $\gamma^\star$. However, when γ goes to $\gamma^\star$, the event $\{S(\boldsymbol{X}) \geq \gamma\}$ becomes more rare, and consequently the CMC estimator

$$\ell(\gamma) = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1}_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \text{ where } \boldsymbol{X}_i \sim f(\boldsymbol{X}) \tag{5}$$

has a relative error

$$RE_{CMC}(\ell(\gamma)) = \frac{\sqrt{1 - \ell(\gamma)}}{\sqrt{C\ell(\gamma)}} \tag{6}$$

that increases to infinity. In order to reduce this relative error, we should increase the sample size C, but then, we would be faced with a computationally intractable problem. Moreover, when γ goes to $\gamma^\star$, it becomes increasingly more difficult to produce samples from $f(\boldsymbol{X})$ that would be close to $\boldsymbol{X}^\star$.

To address this problem, a technique called generalized splitting [4] and derived from Diaconis, Holmes and Ross researches [9] on MCMC (Markov chain Monte Carlo) allows us to compute $\ell(\gamma)$ in an easier and more precise way. For an optimization problem, we will find at least one solution that maximizes our criteria among all the solutions sampled to compute our probability of interest.

If we define a sequence of increasing thresholds γ, such that $\gamma_0 \geq \gamma_1 \geq \ldots \geq \gamma_L$ (with $\gamma_L \leq \gamma^\star$), we can rewrite $\ell(\gamma)$ as the following product of conditional probabilities:

$$\ell(\gamma) = \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_l | S(\boldsymbol{X}) \geq \gamma_{l-1}\right) = c_0 \prod_{l=1}^{L} c_l \ . \tag{7}$$

where

$$c_l = \mathbb{P}_{g^\star_{l-1}}(S(\boldsymbol{X}) \geq \gamma_l) \ . \tag{8}$$

and where the importance sampling density [16]

$$g^\star_{l-1}(X; \gamma_{l-1}) = \frac{\mathbb{1}_{\{S(\boldsymbol{X}) \geq \gamma_{l-1}\}} f(\boldsymbol{X})}{\ell(\gamma_{l-1})} \tag{9}$$

is precisely the conditional density of $\boldsymbol{X}$, given that $S(\boldsymbol{X}) \geq \gamma_{l-1}$. It is worth noting that the support of this density $g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1})$ is precisely the set $\{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) \geq \gamma_{l-1}\}$.

If we know how to draw independent and identically distributed random variables $\boldsymbol{X}_i$ over $\mathcal{X}_{l-1} \in \mathcal{X}$ from this importance sampling function, $\ell(\gamma)$ can be rewritten:

$$\ell(\gamma) = \mathbb{P}_f\left(S(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_{g^\star_{l-1}}\left(S(\boldsymbol{X}) \geq \gamma_l\right). \tag{10}$$

With a judicious choice or a fair estimation of the $\{\gamma_l\}$ sequence, the event $\{S(\boldsymbol{X}) \geq \gamma_l\}$ is no longer a rare event (generally, $c_l \in \left[10^{-3}, 10^{-2}\right]$) under the distribution $g^\star_{l-1}(\boldsymbol{X}, \gamma_{l-1}, \mathcal{C})$ and therefore the c_l quantities can now be well approximated through a CMC estimator. Hence, a CMC estimator of $\ell(\gamma)$ is:

$$\widehat{\ell}(\gamma) = \prod_{l=0}^{L} \widehat{c}_l, \tag{11}$$

where $\widehat{c}_l = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1}_{\{S(\boldsymbol{X}_i) \geq \gamma_l\}}$ and where $\boldsymbol{X}_i \sim g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1})$.

4 Solving Our Real-World Problem

4.1 Evaluating the Detection Probability

According to what we have explained in the first part of the article, we want to maximize the detection probability of a target until time T. This quantity is denoted by $S_T(\boldsymbol{X})$ and is given by the following equation:

$$S_T(\boldsymbol{X}) \triangleq \int_{Y_T \in \mathcal{Y}} f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right) p\left(\boldsymbol{Y}_T | \mu_T(\boldsymbol{X}); \mathcal{C}\right) \, d\boldsymbol{Y}_T \ . \tag{12}$$

Here, $f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right)$ is a cookie-cutter cost function that takes the value 1 if the studied trajectory $\boldsymbol{Y}_T$ satisfies some defined criteria (such as a number of detections and a number of avoidances) and 0 otherwise, *i.e.* $f\left(\boldsymbol{Y}_T|\varphi_T(\boldsymbol{X});\mathcal{C}\right) = \mathbb{1}_{\{\boldsymbol{Y}_T\in A(\boldsymbol{X},\mathcal{C})\}}$ where $A(\boldsymbol{X},\mathcal{C})$ is the set of trajectories which are detected by the solution $\boldsymbol{X}$ and which respect the constraints $\mathcal{C}$. It depends on the visibility of the solution $\varphi_T(\boldsymbol{X})$. $p\left(\boldsymbol{Y}_T|\mu_T(\boldsymbol{X});\mathcal{C}\right)$ is the conditional pdf used to generate the target trajectories and depends on the target intelligence $\mu_T(\boldsymbol{X})$. As this is not the goal of this article, we do not give any more information on how we have implemented this cost function. Unfortunately, $S_T(\boldsymbol{X})$ is an integral with respect to the probability distribution of the (random, solution-dependant) target trajectory $\boldsymbol{Y}$ and its analytical expression is not available. A first approach should be to use the crude Monte Carlo method to obtain an unbiased estimator of $S_T(\boldsymbol{X})$, $\widehat{S}_T(\boldsymbol{X})$:

$$\widehat{S}_T(\boldsymbol{X}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{Y}_T^i|\varphi_T(\boldsymbol{X});\mathcal{C}), \text{ where } \boldsymbol{Y}_T^i \sim p(\boldsymbol{Y}_T|\mu_T(\boldsymbol{X});\mathcal{C}). \tag{13}$$

The trajectories $\boldsymbol{Y}_T^i$ are recursively generated using a first order motion state equation [15] as defined in [7]. To be concrete, we generate a large number of feasible trajectories $\boldsymbol{Y}_T^i, i = 1,\ldots,N$ and evaluate $f\left(\boldsymbol{Y}_T^i|\varphi_T(\boldsymbol{X});\mathcal{C}\right)$.

Note that the relative error associated with $\widehat{S}_T(\boldsymbol{X})$ given by the CMC estimator is

$$RE_{CMC}(\widehat{S}_T(\boldsymbol{X})) = \frac{\sqrt{1-S_T(\boldsymbol{X})}}{\sqrt{NS_T(\boldsymbol{X})}} \;. \tag{14}$$

Also remark that the smaller the probability to estimate, the larger the relative error. To reduce this error, we have to increase the number of trajectories N. Knowing the probability we are meeting is above 10^{-3} (if they were below, planning would be useless), we have chosen $N \geq 50000$.

4.2 The Splitting Algorithm

To solve our problem, a customized version of the splitting method is used. The algorithm we apply is called generalized splitting for research efforts scheduling and is detailed below. For the sequel, we define the function $q(.,\mathcal{C})$ which plays the role of $f(.)$ in the simple case. To begin the computation, we generate an initial pool of feasible solutions with $q(.;\mathcal{C})$. Since a solution and the carrier trajectory are closely linked, we use our trajectory generator to obtain a pool of initial solutions that respect the whole constraints set $\mathcal{C}$.

To ensure our algorithm will not converge and stay into a local extremum, we developed a simple heuristic. If the current maximum score and the current threshold do not increase for a chosen number of times, we automatically reduce the value of the threshold. Through the decrease of the threshold, we start again to accept the feasible solutions generated by the moves and therefore, we reintroduce some diversity in the pool of solutions.

Algorithm 1. The GSRES algorithm

Given parameter ρ, sample number C and number of burn-in iterations b_l of the Gibbs sampler, follow the forthcoming steps:

```
1: Initialization. Set a counter $l = 1$. Generate $C$ feasible solutions $\{\boldsymbol{X}_i\}, i = 1, \ldots, C$ and denote $\mathcal{X}_0$ the set containing them. Note that $\boldsymbol{X}_i \sim q(\boldsymbol{X}; \mathcal{C})$. Evaluate scores $\mathcal{S}_0 = \{\widehat{S}_T(\boldsymbol{X}_i)\}$ and sort in decreasing order $\mathcal{S}_0$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_0 = \widehat{S}_T(\boldsymbol{X}_{j(C_0)})$ with $C_0 = \lfloor \rho C \rfloor$. Define $\widetilde{\boldsymbol{X}}_0 = \widehat{\boldsymbol{X}}_{0:0} = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{\gamma}_{0:0} = \widehat{S}_T(\boldsymbol{X}_{j(1)})$.
2: Selection. Let $\widetilde{\mathcal{X}}_{l-1} = \{\widetilde{\boldsymbol{X}}_1, ..., \widetilde{\boldsymbol{X}}_{C_{l-1}}\}$ be the subset of the population $\{\boldsymbol{X}_1, ..., \boldsymbol{X}_C\}$ for which $\widehat{S}_T(\widetilde{\boldsymbol{X}}_i) \geq \widehat{\gamma}_{l-1}$. $\widetilde{\mathcal{X}}_{l-1}$ contains $\rho$% of the population. Notice that $\widetilde{\boldsymbol{X}}_i \sim g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1}, \mathcal{C})$ for $i = 1, \ldots, C_{l-1}$.
3: Repopulation. Apply one of these methods:
    – Bootstrapping: sample uniformly with replacement $C$ times from the population $\widetilde{\mathcal{X}}_{l-1}$ to define the temporary set of $C$ solutions $\mathcal{X}^{boot}_{l-1}$.
    – ADAM Cloning: make $\left\lfloor \frac{C}{C_l} \right\rfloor + B_i (i = 1, \ldots, C_l)$ copies of each population sample $\widetilde{\mathcal{X}}_{l-1}$. Here each $B_1, \ldots, B_{C_l}$ are $Ber(1/2)$ random variables conditional on $\sum_{i=1}^{C_l} B_i = C \mod C_l$. We then define the temporary set of $C$ solutions $\mathcal{X}^{clon}_{l-1}$.
4: Gibbs sampler. Apply a random Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1}; \mathcal{C}) = \frac{1}{C_{l-1}} \sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i; \mathcal{C})$ with $b_l$ burn-in iterations and the transition density $\kappa_{l-1}$ to each sample of $\mathcal{X}^{boot/clon}_{l-1}$ (see section 4.3) to obtain $\mathcal{X}_l = \{\boldsymbol{X}_i\}$ such that $\boldsymbol{X}_i \sim g^\star_{l-1}(\boldsymbol{X}; \widehat{\gamma}_{l-1}, \mathcal{C})$ for $i = 1, \ldots, C$. Notice that the $\boldsymbol{X}_i, i = 1, \ldots, C$ should be approximately iid.
5: Estimation. Evaluate scores $\mathcal{S}_l = \{\widehat{S}_T(\boldsymbol{X}_i)\}$, $\boldsymbol{X}_i \in \mathcal{X}_l$. Sort in decreasing order $\mathcal{S}_l$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(C_l)})$ with $C_l = \lfloor \rho C \rfloor$. Deduct that $\widetilde{\boldsymbol{X}}_l = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(1)})$, $\widehat{\boldsymbol{X}}_{0:l} = \widetilde{\boldsymbol{X}}_l$ if $\widetilde{\gamma}_l > \widehat{\gamma}_{0:l-1}$, else $\widehat{\boldsymbol{X}}_{0:l} = \widehat{\boldsymbol{X}}_{0:l-1}$ and $\widehat{\gamma}_{0:l} = \max\{\widetilde{\gamma}_l, \widehat{\gamma}_{0:l-1}\}$.
6: Stopping condition. If one of the stopping condition is reached, stop the algorithm and give $\widehat{\boldsymbol{X}}_{0:l}$ as an estimator of the optimal solution. Else $l = l + 1$ and go back to step 2.
```

4.3 The Dedicated Gibbs Sampler

For our problem, we use a *random* Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1}) = \frac{1}{C_{l-1}} \sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i)$ with the transition density κ_{l-1} defined by:

$$\kappa_{l-1}\left(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i\right) = \sum_{j=1}^{6} \lambda_j \prod_{r=1}^{b_l} m_j\left(\boldsymbol{X}_i^r|\widetilde{\boldsymbol{X}}_i^{-r}\right) \ , \tag{15}$$

Here, $\boldsymbol{X}_i^r$ denotes the component r of a solution and $\boldsymbol{X}_i^{-r}$, all the components of $\widetilde{\boldsymbol{X}}_i$ excluding r. The λ_j are the probabilities of updating one component at a time, given that $\sum_j \lambda_j = 1$ and the m_j are the conditional pdf associated to the 6 moves defined in [7].

The random Gibbs sampler will randomly update b_l times the components of a solution $\widetilde{X}_i$. b_l varies during the simulation in this way: $b_l = b_0 + \alpha l$ where $\alpha \in \mathbb{R}_+^\star$. For the first iterations, $b_l < P$ and therefore this approach is faster than a systematic Gibbs sampler. On the contrary, when l is close to L, $b_l \geq P$. Thus, we do more updates than a systematic Gibbs sampler would do but we maintain more diversity in our solutions.

Since we do not know how to update a solution in a way that still satisfies the constraints $\mathcal{C}$, we first recursively propagate the modifications starting from the sensor/activation we have modified in the sequence of activations. Then we check its feasibility, that is, if it respects all the spatial and temporal constraints $\mathcal{C}$. We apply acceptance–rejection (for a limited amount of times) to each updated component until we find a feasible solution. Considering that the cost function S also verifies the consistency of a solution, an updated solution $\boldsymbol{X}_i$ from $\widetilde{\mathcal{X}}_{i,l-1}^{boot/clon}$ is then accepted with probability $\mathbb{1}_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{l-1}\}}$.

Before we proceed further, let us introduce and recall a few notations. $\boldsymbol{s}_i \triangleq [s_{i_x}; s_{i_y}]^T$ denotes the ith sensor position (and more generally the ith sensor), P is the number of sensors in the current solution, P_{max} is the maximum number of sensors, np_i stands for the activations' number for sensor i while $t_{i,\{1,\ldots,np_i\}}$ and $\boldsymbol{\tau}_i$ respectively are the instants of activation of sensor i and the set of activation times associated with sensor i. Also denote by $t_{\boldsymbol{s}_i}$ the set up duration of the sensor $\boldsymbol{s}_i$. Remark that a sensor whose instants of activation are negative is considered as disabled. Consequently, deleting an instant of activation consists of assigning a negative value to this instant. Removing a sensor is then equivalent to deleting all of its instants of activation and ignoring it. Below are the details of the six moves.

1. **Add a sensor.** Sample a position $\boldsymbol{s}'_{P+1}$ from $\mathcal{U}(\Omega; \mathcal{C})$ for the new sensor. Then draw its first instant of activation $t'_{P+1,1} \sim \mathcal{U}([t_{\boldsymbol{s}_{P+1}}, T])$.
2. **Add an instant of activation.** First, choose a sensor randomly *i.e.* draw j uniformly in $\{1, \ldots, P\}$. If $np_j < np_{max}$ then draw $t'_{j,np_j+1} \sim \mathcal{U}([t_{j,1}, T])$.
3. **Remove a sensor.** To apply the move m_3, we apply the following steps : choose a sensor randomly, *i.e.* draw j uniformly in $\{1, \ldots, P\}$. Then delete all of its instants of activation and mark it as disabled
4. **Remove an instant of activation**. Choose a sensor randomly *i.e.* draw j uniformly in $\{1, \ldots, P\}$. We assume that $np_j > 1$. Choose an instant of activation $t_{j,k}$, *i.e.*, draw k uniformly in $\{2, \ldots, np_j\}$. Delete $t'_{j,k}$.
5. **Move a sensor.** Select a sensor $\boldsymbol{s}_j$ randomly, *i.e.* draw j uniformly in $\{1, \ldots, P\}$. Then, draw $\boldsymbol{s}'_j \sim \sum_{k=1}^{2} w_k \, \mathcal{N}(\boldsymbol{s}_j, \boldsymbol{\Sigma}_k^2)$ with $\sum_{k=1}^{2} w_k = 1$. Notice that the weights w_k may evolve during the optimization in order to promote one of the move versus the other. For this mixture of two Gaussian pdf the covariance of the first Gaussian defines a small move while the covariance of the second Gaussian defines a larger move.
6. **Swap two sensors.** If we assume there are at least 2 active sensors, select two sensors $\boldsymbol{s}_k$ and $\boldsymbol{s}_r$ with k uniformly drawn in $\{1, \ldots, P\}$ and r uniformly drawn in $\{1, \ldots, P\} \setminus \{k\}$. For all $k = 2, \ldots, np_k$, delete $t'_{k,j}$ and $t'_{r,j}$ for all $r = 2, \ldots, np_r$. Swap their first instant of activation: $t'_{k,1}$ and $t'_{r,1}$.

5 Illustrative Example: The *Flaming Datum* Search Problem

The first result we present here concerns a scenario in which a target is running away from the position where it has just been detected. Its initial position is drawn from a Gaussian law centred on $\Omega/2$ and with a variance σ^2_{target}. Moreover, the target is supposed to be smart and reactive and therefore, while it is running away, it tries to avoid being detected another time. Considering that the search starts with a delay of t_c^{aoz} which represents the time of arrival of the hunter, we aim to maximize the chances to detect the target during the time T. We use $P_{max} = 10$ sensors that are able to ping only once. For this simulation, we use $C = 800$ solutions, $N = 70000$ trajectories, $b_0 = 2$, $b_l = b_0 + 0.2\ l$ and decide to keep 10% of elites ($\rho = 0.1$). We also let the algorithm perform up to 50 iterations. Because our algorithm is not able yet to adjust the number of sensors considering the cost of their deployment, we have chosen to work with a constant number of sensors. However, we have allowed the removal of a sensor if it is directly followed by an addition of a new sensor. We have used two of the six moves we have defined above : move a sensor and a combination of removing a sensor followed by the addition of a new sensor. The probability of each move to occur is 0.5.

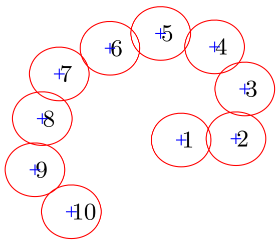

Fig. 2. Graphic of $\boldsymbol{X}^{\dagger}$: position and activation order of the 10 sensors. The red circles delimit the sensors' detection range.

In the best solution we obtain, the sensors position and activation describe a spiral. This result, illustrated in figure 2, is related to the studies of Washburn [12,18] and Son [17] for an only-spatial optimization case, *i.e.*, when the target is not able to avoid the sensor ("myopic" case). In this context, the best spatial sensor deployment designs an Archimedean spiral. Note that as the algorithm reaches the 30^{th} iteration, it has almost converged and the solution resembles a spiral. Figure 3 shows 3 steps of the simulation for the best solution found. The green crosses (+) represent non detected targets, the orange stars represent warned/avoiding targets and the red crosses (x) represent detected targets. The red circles delimit the sensors' detection range and the orange circles delimit the target avoidance zone.

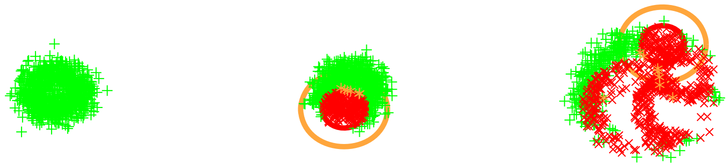
Fig. 3. 3 steps of the simulation for the best solution found. (a) Beginning of the simulation, the targets are not detected. (b) Activating of the first sensor. (c) Activation of the fourth sensor.

Continuing, we plot the optimization evolution behaviour versus iterations $\widehat{\gamma}_{0:l} = \widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ which represents the mean score of the current population (figure 4). Both are smoothly increasing in a logarithmic way. On figure 5 we see that the support of scores pdf is large at the beginning ($l = 0$) but becomes thinner and converges towards a Dirac pdf as the optimization is conducted. Moreover, $\widehat{\gamma}_{0:l}$ increases and the standard deviation of the distribution decreases. Once the optimization is over, we observe that the detection probability reaches 0.9406 whereas when $l = 0$, the best score is below 0.25 and the mean scores $\mathbb{E}\left[\widehat{S}_T(\boldsymbol{X})\right]$ is equal to 0.0187 with a large standard deviation. This gap illustrates the efficiency of our approach, which also gives us good results with other types of scenarii [14].

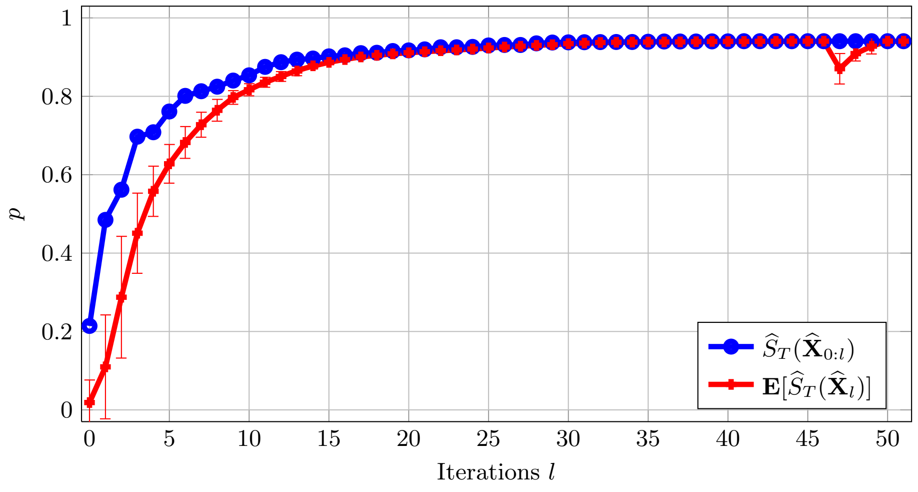

Fig. 4. In blue $\widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and in red $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ with $C = 800$, $N = 70000$, $L = 50$

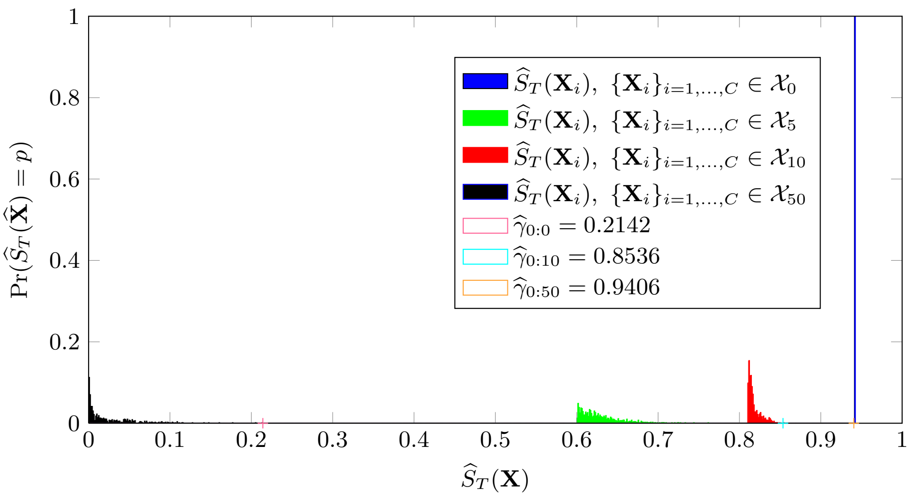

Fig. 5. Scores densities support versus iterations GSRES. In black $l = 0$, in green $l = 5$, in red $l = 10$ and in blue $l = 50$.

6 Conclusion and Prospects

In this work, we have presented an approach based on the rare events simulation framework and the generalized splitting algorithm. We have shown that this method is very similar to non-parametric genetic algorithms. This method has been applied to a strongly constrained optimization problem and tested with the flaming datum scenario.

The next step is to take the cost of each solution into account in order to extend our algorithm to the multi-objective case. To achieve this, we shall develop a method based on a Pareto-ranking algorithm. Although using a Choquet integral [10] may also be a good choice, it requires much more information from the decision maker. Furthermore, the Pareto-ranking method [2] has already been used with evolutionary algorithm with success.

Acknowledgement. This work was partially supported by DGA (Direction générale de l'armement). All four authors gratefully acknowledge Emile Vasta (DGA/Techniques navales) for his friendly support and interest in this work.

References

1. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evol. Comput. 1, 1–23 (1993), `http://dx.doi.org/10.1162/evco.1993.1.1.1`

2. Bekker, J., Aldrich, C.: The cross-entropy method in multi-objective optimisation: An assessment. European Journal of Operational Research 211(1), 112–121 (2011)
3. de Boer, P.T., Kroese, D., Mannor, S., Rubinstein, R.: A tutorial on the cross-entropy method. Annals of Operations Research 134(1), 19–67 (2005), `http://dx.doi.org/10.1007/s10479-005-5724-z`
4. Botev, Z., Kroese, D.: An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting. Methodology and Computing in Applied Probability 10(4), 471–505 (2008), `http://dx.doi.org/10.1007/s11009-008-9073-7`
5. Boubezoul, A., Paris, S., Ouladsine, M.: Application of the cross entropy method to the GLVQ algorithm. Pattern Recogn. 41, 3173–3178 (2008), `http://portal.acm.org/citation.cfm?id=1385702.1385950`
6. Bouzarkouna, Z., Auger, A., Ding, D.Y.: Investigating the Local-Meta-Model CMA-ES for Large Population Sizes. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 402–411. Springer, Heidelberg (2010), `http://hal.archives-ouvertes.fr/hal-00450238/en/`
7. Chouchane, M., Paris, S., Le Gland, F., Ouladsine, M.: Splitting method for spatio-temporal search efforts planning (May 2011), `http://arxiv.org/abs/1105.3351v1`
8. Dell, R.F., Eagle, J.N., Alves Martins, G.H., Garnier Santos, A.: Using multiple searchers in constrained-path, moving-target search problems. Naval Research Logistics 43(4), 463–480 (1996)
9. Dianonis, P., Holmes, S.: Three examples of Monte-Carlo Markov chains. Discrete Probability and Algorithms, 43–56 (1994)
10. Grabisch, M.: L'utilisation de l'int'egrale de Choquet en aide multicritère à la décision. Newsletter of the European Working Group "Multicriteria Aid for Decision" 3(14), 5–10 (2006)
11. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, pp. 312–317. Morgan Kaufmann (1996)
12. Hohzaki, R., Washburn, A.: The diesel submarine flaming datum problem. Military Operations Research 6(4), 19–30 (2001)
13. Akbari, R., Ziarati, K.: A multilevel evolutionary algorithm for optimizing numerical functions. International Journal of Industrial Engineering Computations 2, 419–430 (2011)
14. Rodrigues, C., Michelon, P., Quadri, D.: Un modèle bi-niveau pour le problème de la recherche d'une cible dynamique. MajecSTIC 2009 (2009)
15. Rong Li, X., Jilkov, V.P.: Survey of maneuvering target tracking. Part i. Dynamic models. IEEE Transactions on Aerospace and Electronic Systems 39(4), 1333–1364 (2003), `http://dx.doi.org/10.1109/TAES.2003.1261132`
16. Rubinstein, R.Y.: The Gibbs cloner for combinatorial optimization, counting and sampling. Methodology and Computing in Applied Probability 11(4), 491–549 (2009)
17. Son, B.: Tracking Spacing for an Archimedes Spiral Search by a Maritime Patrol Aircraft (MPA) in Anti-submarine Warfare (ASW) Operations. Master's thesis, Naval Postgraduate School (December 2007)
18. Washburn, A.: Search and Detection. INFORMS (2002)
19. Washburn, A.: Branch and bound methods for a search problem. Naval Research Logistics 45(3), 243–257 (1998)