Genetic regulatory networks inference : modeling, parameters estimation & model validation

Fomekong Nanfack, Y.

Publication date
2010

3

Hybrid (μ, λ)-evolution strategy and Direct Search[1]

In the previous chapter, we have discussed general aspects of parameter estimation including different methodologies and their issues. Undeniably, a need for efficient methods for model calibration of complex dynamical biological systems, and particularly models of pattern formation is necessary. In this method chapter, we present an efficient approach to estimate unknown parameters in nonlinear dynamic multi-dimensional models of pattern formation.

The method presented here combine a global search based on evolutionary strategy (ES) followed by a local search. Our choice is inspired by [175, 184] where the authors compared different global optimisation strategies and suggested that the modern evolution strategy is the most competitive and the only one capable of finding the true minimum in the parameter estimation of biochemical networks. Their study focuses on the inverse problem of a benchmark model of a three-step pathway with 36 unknown parameters. Our contribution extends the applicability of ES to higher multi-dimensional non-linear model of genetic regulatory network that have been so far inferred using simulated annealing strategies. Our primarily motivation is to reduce the computational effort of the previous used method in similar problems. Second, if the computational time is reduced, our goal is to obtain more results in order to have more depth analysis of the in-silico GRNs.

[1] This chapter is partially based on the paper:
Yves Fomekong-Nanfack and Jaap A. Kaandorp and Joke Blom, "Efficient parameter estimation for spatio-temporal models of pattern formation: Case study of *Drosophila melanogaster*", Bioinformatics, 23(24): 3356 - 3363, 2007. [78]

Section 3.1 describes the basic ES and the stochastic ranking ES. Section 3.2 presents an island-ES. The local search is presented in section 3.3.

3.1 (μ, λ)-evolution strategy

Evolution strategy is an Evolutionary Algorithm (EA) that belongs to the class of algorithms presented in section 2.2.2.2. Like all EAs methods such as Genetic Algorithm (GA) or Evolutionary Programming (EP), it is a stochastic iterative algorithm that operates on some encoded individuals from an initial population. Historically, GA became popular for discrete, combinatorial search after Holland's book "Adaptation in Natural and Artificial Systems" in 1975 [110]. EP, first introduced by L.J. Fogel [77] is used for continuous parameter optimisation. The origin of ES is from the early 1960s by Ingo Rechenberg [219] and popularised by Hans-Paul Schwefel [244]. It was conceived as a set of rules for the automatic design and analysis of consecutive experiments with stepwise variable adjustments driving a suitably flexible object/system into its optimal state under noisy and multimodal conditions [21]. Later on, ES was not only applied to discrete problems but also to continuous decision variables.

ES, like most EAs consists of three main operators: crossover, mutation and selection. The first two are exploration operators of the search space, while the last one lets the population evolve towards the optima of a problem. The performance of ES critically depends on the adjustment of the internal parameters, with a main dependence on the mutation strength. One of the main advantages of ES compared to standard EAs is the usage of strategic parameters such as on-the-fly adaptation of the mutation parameters. Many different specialised version of ES exist, but the two most popular and successful setting of ES are:

1. $(\mu + \lambda) - ES$. In a population of λ individuals, $\mu + \lambda$ new offsprings are created at each generation. To keep the population size constant, only the best between the parents and the children are kept.

2. $(\mu, \lambda) - ES$. Only the best μ out of the λ parents are chosen to create a new population.

The symbol μ designates the number of parents chosen in a population to be the genitors and the symbol λ represents the number of offsprings created by these parents. In both algorithms, λ offspring individuals are generated from μ parents. The difference lies only in the fact that in the ÒplusÓ strategy, μ parents for the next generation are selected from a combination of the μ parent and λ offspring individuals of this generation, whereas in the ÒcommaÓ strategy, μ parents for the next generation are selected from the λ offspring individuals in this generation only.

In this study we use a modified (μ, λ)-ES, based on stochastic fitness ranking. This method is simple and has proven to be more efficient than most EAs and

ESs for large parameter estimation problems [233, 234]. A more rigorous description of the (μ, λ)-ES is as follows:

$$(\mu, \lambda) - ES = (\mathbb{P}; \mu; \lambda; \mathbf{sel}; \mathbf{rec}; \mathbf{mut}; \sigma; V, G, t) \tag{3.1}$$

where:

- $\mathbb{P} = (I_1, ...I_\lambda) \in \mathbb{I}^\lambda$. I_j is an individual described in Equation 3.2
- $\mu \in \mathbb{N}$ is the number of parents selected at each generation
- $\lambda \in \mathbb{N}$ is the population size and the number of offsprings ($\lambda > \mu$)
- $\mathbf{sel} : \mathbb{I}^\lambda \longrightarrow \mathbb{I}^\mu$ is the selection operator
- $\mathbf{rec} : \mathbb{I}^3 \longrightarrow \mathbb{I}$ is the recombination operator
- $\mathbf{mut} : \mathbb{I} \longrightarrow \mathbb{I}$ is the mutation operator
- $\sigma \in \mathbb{R}^\lambda \times \mathbb{R}^N$ is the step size meta-control
- $V : \mathbb{R}^N \longrightarrow \mathbb{R}$ is the objective function
- $G : \mathbb{R}^{N.q} \longrightarrow \mathbb{R}^q$ is the constraint function with q the number of constraints
- $t : \mathbb{I}^\mu \longrightarrow \{0, 1\}$ is the termination criterion

Each individual is a representation of the parameter vector plus a set of endogenous parameter:

$$I_j := (\theta_j, \sigma_j, V_j, G_j), \forall \quad j \quad \in \quad \{1, ...\lambda\} \tag{3.2}$$

where $\theta_j := (\theta_j^1, ...\theta_j^N)$ is the parameter set vector and $\sigma_j := (\sigma_j^1, ...\sigma_j^N)$ the endogenous strategy parameter vector associated to each parameter of each individual. n is the problem dimension. V_j is the objective function and G_j is the penalty function. The parameter vector σ is used to control statistical property during the evolution of the algorithm. Initialisation consist in generating randomly λ number of individuals. Each individual is generated according to a uniform N-dimensional probability distribution over the search space. The following paragraphs describe the different main operators.

3.1.1 Selection by Stochastic Ranking

In section 2.2.5 of Chapter 2, we have discussed the issues of constraints in optimisation problem. In most cases, constraints are defined as a strict space boundary on the parameters such as: $\theta_i \in [\theta_i^l, \theta_i^u]$ where θ_i^l and θ_i^u are respectively the lower and upper bound. It is also common that beside parameter restriction, additional constraints are imposed on the minimisation of the function V. These constraints can be defined by simplicity as penalty function method and need to be handle in the optimisation algorithm. Inclusion of the

penalty tends to transform the constraint optimisation to an unconstraint one. The reformulation of the objective function given in Equation 2.4 is :

$$\begin{aligned} V(x, \boldsymbol{\theta}) &= V_{MLE}(\boldsymbol{\theta}) + w_0 \sum_{j=1}^{m} w_j (g_j^+(\boldsymbol{\theta}))^\beta \\ &= V_{MLE}(\boldsymbol{\theta}) + w_0 \mathbf{G}(g^+(\boldsymbol{\theta})) \end{aligned} \tag{3.3}$$

where $\mathbf{G}(g^+(\boldsymbol{\theta}))$ is the total penalty function. Each function g_j is a penalty expressing a particular constraint with a given weight w_i and β exponent. The weight w_0 is the coefficient that gives the importance of the overall constraint. However, it is in practice very difficult to find the optimal value of the weight, especially for w_0. It is necessary to find an efficient manner to balance between the objective function $V_{MLE}(\boldsymbol{\theta})$ and the constraints violation. To avoid the setting of the weight penalty parameters in Equation 3.3 , Runrasson and Yao [233] suggested to handle the penalty functions as multiobjective optimisation where the constraints are treated as additional optimisation function. The strength of this approach is to neither consider under- nor over-penalisation but a well balance method to preserve individuals that might break the constraints (being in an infeasible region of the search space) but have a good objective.

Let $\mathbf{P_f}$ be the probability of using only the objective function $V_{MLE}(\boldsymbol{\theta})$ for comparisons of ranking in the infeasible regions of the search space. Given two adjacent individuals among the λ's, if both individuals are feasible the probability of comparing them according to their objective function if it is 1, otherwise it is $\mathbf{P_f}$. The total population is ranked using a bubble-sort Algorithm where the conditional swap between two adjacent individuals depends on the previous mentioned probability. The following algorithm describes the stochastic ranking procedure.

Algorithm 1 Bubble sort with stochastic ranking

```
1: I_j  ∀j ∈ {1, ..., j}
2: for i  1  to  N do
3:   for j  = 1  to  λ − 1 do
4:     random number u  ∈ U(0, 1)
5:     if G(I_j) = G(I_{j+1}) = 0 or (u < P_f) then
6:       if V_MLE(I_j) > V_MLE(I_{j+1}) then
7:         swap(I_j, I_{j+1})
8:       else if G(I_j) > G(I_{j+1}) then
9:         swap(I_j, I_{j+1})
10:      end if
11:    end if
12:  end for
13:  IF no swap done break.
14: end for
```

where I_j is the j^{th} individual of the population and u is a uniform random number. The choice of $\mathbf{P_f}$ determines the strength of the penalisation. Weak probability criteria; $\mathbf{P_f} \to 0$ implied stronger penalisation while strong probability threshold $\mathbf{P_f} \to 1$ reduces the influence of the penalisation. Intuitively one will set $\mathbf{P_f} = \frac{1}{2}$ for an equal chance of comparison based on the objective or the penalty. It was shown that $\mathbf{P_f} < \frac{1}{2}$ to achieve better results [233].

3.1.2 Recombination

Also named crossover, recombination is the primary exploration mechanism in most GAs. We use a global intermediate recombination described in Equation 3.4

$$\theta_i' = r(\theta_o, \theta_c, \theta_{i+1}) = \theta_i + \gamma\,(\theta_o - \theta_{i+1}) \tag{3.4}$$

where θ_i is the parameter vector of an individual i, individual o is the highest ranked individual (the fittest), and γ is the recombination factor. In this way a number of μ new individuals is created from the λ offspring. The individuals c are chosen among the best μ offspring obtained after a stochastic ranking [233]. The rest of the new population is filled with the (unchanged) μ best individuals (repeatedly).

3.1.3 Mutation and Self-Adaptation

After the cross-over, mutation is applied to the $\lambda - \mu$ individuals. Mutation is the primary source for individual variation through the evolution. An efficient ES requires a good setting of the mutation coefficient α. It was shown that it is possible to improve the convergence rate by using a variable mutation strength σ instead of a constant. However the mutation coefficient α should evolve in

a very efficient way. It is important to define the mutation strength properly. As discussed in the previous chapter, the mutation operator is usually a basic variation operator in ES although the design of a mutation operator is problem dependent. By analyzing implementation details of various ES, Beyer et al. [20] proposed some rules to design an efficient mutation methodology:

1. reachability: to ensure that starting from a parent, any child can be reached in a finite number of mutation steps or generation.

2. unbiasedness: use selection and variation respectively to 1) exploit the fitness information in order to guide the search into promising search space, 2) explore the search space. The variation operation should not introduce any bias.

3. scalability: the mutation strength or the average length of the mutation step should be tunable in order to adapt the properties of the fitness landscape. The scalability should guarantee evolvability of the ES.

These theoretical considerations can be observed when minimizing an objective function of a sphere model using a simple $(1 + 1) - ES$ with isotropic Gaussian mutations with constant mutation strength $\sigma = 1.0$. In this test case shown by Beyer et al. [20], after a period of good performance the ES loses its evolvability and starts to stagnates as shown in Fig. 3.1. At each iteration, only one parent is selected to generate the next offspring. The progress rate φ and the success probability P_s both depend on the mutation coefficient, but also the mutation strength σ. Theoretical and computational results obtained by Rechenberg et al. [218] and Beyer et al. [20] suggested a compromise success probability of $P_s = 0.2$ leading to the $1/5th$-rule:

"in order to obtain nearly optimal (local) performance of the (1+1)-ES in real-valued search spaces, tune the mutation strength in such a way that the (measured) success rate is about $1/5$*"*.
This rule is restricted to the (1+1)-ES but can be extended to a more general self-adaptation ES. In a self-adaptative ES, the mutated strategy parameters are used to control the mutation coefficient α applied to the individual's object parameters. The strategy parameter σ controls the strength of the object parameters by tuning the search distribution so that maximal progress is maintained (mutations become smaller as the search progresses). Since the strategy parameters are endogenous, it is important to carefully control their evolution. Let us consider the adaptation of a single mutation strength parameter σ. The first step consists in the initialisation of the strategy parameter σ. If we assume that $\delta\theta$ is an estimation of the expected distance to the global minima, according to [246], the initial step-size is given by:

$$\sigma_j^0 = \delta\theta_j/\sqrt{N} \approx (\theta_j^u - \theta_j^l)/\sqrt{N} \tag{3.5}$$

with $j \in \{1, ..N\}$, N being the number of parameters to estimate. The strategy parameter σ represents the standard deviation of the mutation distribution.

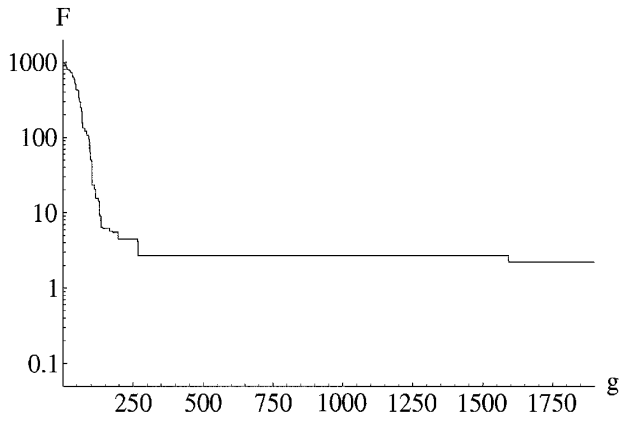

Figure 3.1: A simple (1+1)-ES based on isotropic Gaussian mutations with constant mutation strength σ = 1.0is used to minimize a sphere model. At each iteration, only one parent is selected to generate the next offspring. A quick convergence is observed the first 250 steps. After, the ES shows very little improvement of the fitness and starts to stagnate.

This initial value is used as upper bound on σ. The next step is the mutation of the strategy parameter. The successive mutations should decrease σ until a certain $\epsilon_{sigma} \to 0$. Lets consider the spherical model taken as an example above with one mutation strategy parameter. If we assume that there exits an optimal $\hat{\sigma}$, due to the symmetry of the sphere, $\hat{\sigma}$ depends only on the parental distance r_p to the optimum at a given state p. The state space and the mutation are scaled by a constant factor, both $\hat{\sigma}$ and r_p are scaled in the same way implying that: $\hat{\sigma}/r_p = const$ (under stationary conditions, i.e., self-adaptation is working properly). If we consider two consecutive generations, it follows:

$$\hat{\sigma}^{(g)}/r^{(g)} = \hat{\sigma}^{(g+1)}/r^{(g+1)} \quad \Rightarrow \quad \hat{\sigma}^{(g+1)} = \hat{\sigma}^{(g)} \frac{r^{(g+1)}}{r^{(g)}} \tag{3.6}$$

where g is the current generation. The expected relative rate change should be constant. r and σ should be changed by a constant factor using a log-normal rule:

$$\tilde{\sigma_j} := \sigma \exp(\tau \mathcal{N}(0,1)) \tag{3.7}$$

where τ is an exogenous parameter called the learning rate and $\mathcal{N}(0,1)$ a random number derived from a normal distribution [245]. The learning rate τ has been theoretically and empirically investigated by Schwefel et al. [245] and they suggested that:

$$\tau = 1/\sqrt{N} \tag{3.8}$$

This holds for problems with very smooth fitness landscape. However, for highly multimodal Þtness landscapes, it is suggested to use smaller learning rates such as:

$$\tau = 1/\sqrt{2N} \tag{3.9}$$

The mutation rule given in Equation 3.7 is defined for the case where only one mutation parameter strategy is used. This log-normal rule can be extended to the case where each parameter has its own strategy parameter. In this case, the strategy parameters are given in a vector $\sigma = (\sigma_1, ...\sigma_N)$ and follow an extended log-normal rule given as:

$$\tilde{\sigma} := \exp(\tau_0 \mathcal{N}_0(0,1)).\big(\sigma_1 \exp(\tau \mathcal{N}_1(0,1)), ..., \sigma_N \exp(\tau \mathcal{N}_N(0,1))\big) \tag{3.10}$$

This mutation strategy extends the previous one given in Equation 3.7 by defining a general mutative multiplier with learning rate τ_0 and coordinate-wise mutations with learning parameter τ. Each component of the vector σ is mutated independently as given in Equation 3.10 and the whole vector is mutatively scaled by random factor $\exp(\tau_0 \mathcal{N}_0(0,1))$. Since each parameter for each individual has its own mutation strategy, it can be read as:

$$\sigma'_{i,j} = \sigma_{i,j} \exp\left(\tau_0 \mathcal{N}(0,1) + \tau \mathcal{N}_j(0,1)\right) \tag{3.11}$$

$\mathcal{N}(0,1)$ is a normally distributed uniformly random variable and $\mathcal{N}_j(0,1)$ a new random variable generated for each parameter j. Subscript i is the individual index. $\tau = 1/\sqrt{2\sqrt{N}}$ is the coordinate-wise learning rate of the parameters and $\tau_0 = 1/\sqrt{2n}$ the overall learning rate [246, 247].

Then, the parameters are mutated according to:

$$\theta'_{i,j} = \theta_{i,j} + \sigma'_{i,j} N_j(0,1) \tag{3.12}$$

Equation 3.11 introduces noise on σ by means of random fluctuations. Consequently, the performance of the ES is degraded and although it is impossible to achieve the theoretical maximal progress rate, any reduction of the random fluctuations will be a major improvement [20]. Ostermeier et al. [200] have proposed to reduce random fluctuations by recombining or averaging the strategy parameters. One limitation of this approach is to use strategy parameters of an other individual especially if this individual is located in another region of the search space (typically in Pareto based method). Runrasson [232] offered an alternative method to reduce random fluctuations without averaging over the population, but instead "*takes an exponential recency-weighted average of trial step sizes sampled via the lineage instead of the population*" [234]. Simply speaking, an exponential smoothing is applied to the strategy parameter as follow:

$$\sigma'_{i,j} = \sigma_{k,j,} + \alpha(\sigma'_{i,j} - \sigma_{k,i}) \tag{3.13}$$

with $k = i \mod \mu$ is the index of the strategy parameter of the same lineage. The coefficient α which was intended to be use as classic mutation coefficient for the parameters θ is instead used as the smoothing factor.

3.1.4 Pseudo-code

The following pseudo-code summarises the different steps of the (μ, λ)-ES discussed above.

Algorithm 2 (μ, λ)**-ES**

INITIALIZATION: $\sigma'_{i,j} = (\overline{\theta_j} - \underline{\theta_j})/\sqrt{(n)}$; $\theta'_{i,j} = (\overline{\theta_j} - \underline{\theta_j})\mathbf{U}_j(0,1)$
while termination criteria not met **do**
  SCORE: evaluate each individual objective function: $V_{MLE}(\theta'_i)$ and penalty: $\mathbf{G}(\theta'_i)$
  RANKING: sort individuals based on a stochastic ranking.
  SELECTION: select the μ best individuals out of λ offspring as parents for the next generation.
  COPY μ individuals λ/μ times $\underbrace{\{(\theta_1, \sigma_1); ....; (\theta_\mu.\sigma_\mu)\}}, ..., \underbrace{\{(\theta_1, \sigma_1); ....; (\theta_\mu.\sigma_\mu)\}}$
  for $k = 1 \quad to \quad \lambda$ **do**
    ß = $\mod (k-1, \mu) + 1$
    if $(k < \mu)$ **then**
      recombination
      $\sigma'_{k,j} \leftarrow \sigma_{k,j}$
      $\theta'_{k,j} \leftarrow \theta_{k,j} + \gamma\,(\theta_{1,j} - \theta_{k+1,j})$
    else
      non-isotropic mutation
      $\sigma'_{k,j} \leftarrow \sigma_{i,j} \exp(\tau' N(0,1) + \tau N_j(0,1))$
      $\sigma'_{k,j} \leftarrow \theta_{i,j} + \sigma'_{k,j} N_j(0,1)$
      $\sigma'_{k,j} = \sigma_{i,j,} + \alpha(\sigma'_{k,j} - \sigma_{k,i})$
    end if
  end for
end while

3.2 Island (μ, λ)-evolution strategy

Implicitly, EAs are inherently parallel. A natural process would assume parallel evolution of the individuals, where all the mechanisms or operators such as recombination or mutation would append asynchronously. This parallel description of the algorithm requires a a fine-grained massively parallel computer [288] where, the ideal situation would consist in only one individual per processor. However, due to physical limitation, this approach is not frequently used. The two main alternatives are the global single-population master-slave

EAs and the multipopulation coarse-grained EAs [35]. In the master-slave EA has one single population residing on the master processor where common operations are performed (selection and re-population). Only the fitness assignment is distributed among the slave processors. In the muliple-population EAs, the population is divided into sub-population based on the number of available processors. Each processor runs independently a sequential EA on its own sub-population. After an isolation period, the sub-populations exchange individuals.

Comparisons of the single-population GA and multipopulation GA suggested that the multipopulation has several advantages over the other one and leads to better results, using the same number of individuals [35, 162, 164, 187]. We have therefore use this approach to implement an island-based (μ, λ)-ES where each sub-population evolves independently as described in Section 3.1.4. The three basic and important features of an island-ES are:

1. how often do the islands exchange individuals?
2. how many individuals are exchanged?
3. how do the island exchange their individuals?

The island-ES is defined as follow:

$$(\mu, \lambda) - ILES = ((\mu', \lambda/n_p) - ES; n_p; \kappa; \pi; S;) \tag{3.14}$$

where:

- $(\mu', \lambda/n_p) - ES$ is a independent ES defined in Equation 3.1.
- n_p is the number of sub-population
- κ is the migration interval
- π is the migration rate
- S structure of the migration (or connection between island)

Initialising each island independently ensures a diverse set of individuals covering a large part of the parameter search space. After the isolation time (or migration interval κ), a number of individuals is exchanged among the sub-populations by a procedure called migration. The migration interval should not be too small or neither too large since the method would almost be reduced to a single-population EA in the first case, or to completely independent EA in the second case. The number of exchanged individuals (migration rate π) can be selected randomly or fitness-based. The migration topology determines how do they island exchange individuals. Among many approaches, the most common schemes are complete net structure and ring topology.
The migration operation spreads the best individuals over sub-populations. An elitist migration is applied where only the best individual of each island is

migrate and replace the worst of the target island. This elitist migration ensures that the new individual inserted in a subpopulation can allow the population to escape local minima if trapped in one with a high value of the cost-function. Also, migration allows to maintain diversity in the sub-populations [99, 286]. We use a complete net structure [164] with random assignment. At each migration, an individual from a population can migrate to any other subpopulation. Fig. 3.2 illustrates the migration scheme used for the rest of our work.

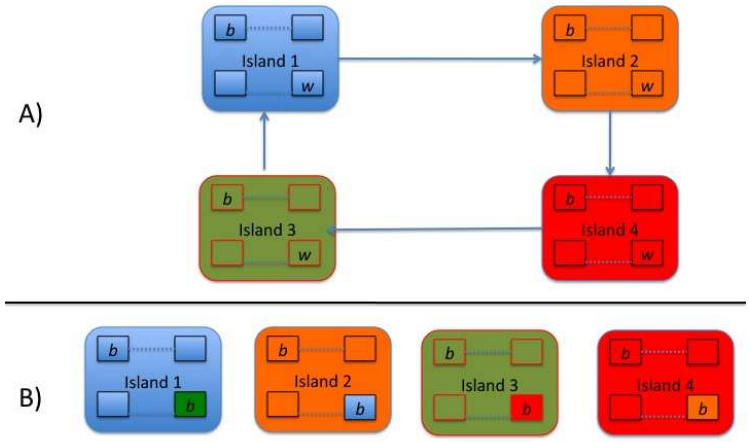

Figure 3.2: Fitness-based complete-migration topology with 4 islands. Each colour represents an island. The little squares represent individuals, where "*b*" stand for individual with the best fitness and "*w*" for the one with the worst fitness. A) the arrows between indicate the migration topology. B) islands after the migration. The worst in each island has been replaced with the best coming from the migrating island.

It has been shown [35,187] that an island evolution algorithm can qualitatively outperform a serial EA. The focus is not on the performance in terms of computational time of a parallel version of ES, but on its effectiveness in terms of the quality of the solution. In the current work, we did not consider a parallel implementation of the ES, but a mimetic multipopulation based ES run in sequential. The island-ES used here is run on a single processor, working as a regional model.

3.3 Local Search

Global search is often used for parameter estimation problems where no information on parameters is available. Although it has proven to be efficient in many problems to identify promising regions, a slow convergence when reaching the global minima is always observed. Combining a global search with a local optimizer to identify the minimum speeds up convergence. The hybrid approach used is inspired from Katare et al. [132] where the authors have successfully used a hybrid genetic algorithm to estimate parameters of small (5 parameters) and large (31 parameters) kinetic model of propane aromatisation on a zeolite. Also, Gursky et al. [102] used a local search to refine the parameters obtained after a global search by simulated annealing.

There are a large variety of local search techniques. Most local optimizer techniques such as Powell's method, the quasi-Newton methods or Levenberg-Marquardt (see Section 2.2.3) are based on the gradient descent approach and thus require the derivative of the objective function $f(\theta)$. If analytic expressions are not available for the derivative, a finite-difference approximation of the gradient of $f(\theta)$ can be used. In many situations, computing the objective function $f(\theta)$ can be expensive and numerical approximation of the gradient of $f(\theta)$ is thus too costly. Furthermore, biological data can be noisy, making the use of the gradient difficult if not impossible. In these cases, Newton-like local optimizers become inappropriate. A good alternative is a direct search method (see Section 2.2.3.1). Direct search such as generating set search [147, 160], pattern search [111] or Downhill simplex [194] are suitable to solve a variety of optimisation problems that are not well suited for standard optimisation algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear. In this study we use the Downhill simplex (DS) as local search strategy. DS assumes that the initial starting point (simplex) is around a local minimum. Simplex-based direct search methods are based on a comparison of the cost-function values at the vertices of a simplex that is updated by the algorithm steps. (A simplex is the geometrical figure consisting, in N dimensions, of $N + 1$ points (or vertices) and all their interconnecting line segments, polygonal faces, etc., giving in 2D a triangle and in 3D a tetrahedron.)

3.4 Conclusions

In this chapter, we have presented the parameter estimation method developed for the purpose of our gene regulatory network inference. The choice of an hybrid approach in many respects is motivated by previous work such as the one by Voogd et al. [281]. Out of all the global search methods, we have chosen an evolutionary strategy based on stochastic ranking strategy which was reported to be one of the more efficient global stochastic search algorithm. This type of algorithm has proven superiority over algorithms like simulated

annealing which indicate that evolution strategies might be the most competitive stochastic optimization method, especially for large problems and nonlinear dynamic systems [220]. Most of the biological problems where stochastic global optimization methods are compared deal with problem with relative low unknown parameters. In the next chapter, we will present a known biological problem for which simulated annealing have been used to infer a GRN with 62 unknown parameters. We will use the hybrid method presented in this chapter to infer the same system of GRN, followed by a comparison and details analysis of our results.