3 Theoretical Background

Finding the best solution to an equation or set of equations, often limited by certain criteria, is the goal of the field of mathematical-optimizations, alternatively mathematical-programming. Such a problem for mathematical-optimization can be categorised into being either discrete, with countable sets of solutions, or continuous, inversely with uncountable/continuous sets of solutions. Typically these limitations, more commonly known as constraints, can be considered sufficiently with continuous optimization problem.

Optimization has a wide range of application, from engineering to chemistry, from transportation to biophysics, it finds its uses. Usual formulation of the problem is in terms of minimization or maximization, hence the readily application of optimization as a means to procure parameters for biological processes, they too are subjected to, i.e. constraint by, an environmental and evolutionary pressure.

3.1 Problem Definition

The optimization algorithms are tasked to calculate the solution to the equation defined in this section. The acquired solutions represent the impact of a TFs on its target, or of a phosphorylation site (PSite) on its TF.

Before getting into the construction of the optimization problem, here on just named Problem, a few variables are necessary to be defined. For a mRNA i the vector of its measured time series is defined as $\vec{R}_i$, the corresponding time series vector of the jth TF is denoted with $\vec{TF}_{i,j}$. This TF vector does not include the PSite fold-changes, those are gathered in the matrix $\mathbf{P}_j$, note that the first column does not represent the fold-change at time point zero. The aforementioned inhibition or activation is facilitated in the model through α-values for the TFs and β-values for the PSites. All α-values effecting the TFs of a mRNA i are gathered in the vector $\vec{\alpha}_i$, likewise the β-values acting on the PSites of a TF j are in the vector $\vec{\beta}_j$. Both $\vec{\alpha}_i$ and $\vec{\beta}_j$ are constant. Additionally the vector $\vec{\beta}_j$ is always the same regardless where its related TF is involved. The value β^0 included in $\vec{\beta}_j$ does not effect a PSite, only the TF directly, with a side effect of including possibly undetected PSites.

$$\vec{R}_i = \{[mRNA]_i(t_1), \ldots, [mRNA]_i(T)\} \tag{3.1}$$

$$\vec{TF}_{i,j} = \{[TF]_{i,j}(t_1), \ldots, [TF]_{i,j}(T)\} \tag{3.2}$$

$$\mathbf{P}_j = \begin{Bmatrix} 1, & [PSite]_j^1(t_1), & \ldots & [PSite]_j^k(t_1) \\ \vdots & \vdots & & \\ 1, & [PSite]_j^1(T), & \ldots & [PSite]_j^k(T) \end{Bmatrix} \tag{3.3}$$

$$\vec{\alpha}_i = \{\alpha_{i,j}, \ldots, \alpha_{i,m}\} \tag{3.4}$$

$$\vec{\beta}_j = \{\beta_j^0, \beta_j^1, \ldots, \beta_j^k\} \tag{3.5}$$

$$\tag{3.6}$$

Furthermore the parameters are restricted in their magnitude. Each $\alpha_{i,j}$ can only take values between zero and one, β_j^q is constrained between negative two and two. The data used for this work are fold change values, thus without a unit and defined at time point zero to have the value one, which leads to the following constraints

$$0 \leq \alpha \leq 1 \tag{3.7a}$$

$$-2 \leq \beta \leq 2 \tag{3.7b}$$

$$\sum_{j=1}^{m} \alpha_{i,TF_j} = 1 \tag{3.7c}$$

$$\sum_{q=1}^{k} \beta_j^q = 1 \tag{3.7d}$$

$$\tag{3.7e}$$

With the core model equation for each mRNA in the network defined as

$$\vec{R}_i = \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i, \tag{3.8}$$

the problem is defined. To give a more comprehensible presentation of the core equation an expanded equation is provided below in (3.9). This example is the equation for a mRNA on which two TFs act, the first with two P-Sites and the other with one P-Sites:

$$R_i(t_1) = \alpha_{i,TF_1} \cdot TF_{i,1}^0(t_1) \cdot (\beta_1^0 + PSite_{i,1}^1(t_1) \cdot \beta_1^1 + PSite_{i,1}^2(t_1) \cdot \beta_1^2) \tag{3.9}$$

$$+ \alpha_{i,TF_2} \cdot TF_{i,2}^0(t_1) \cdot (\beta_2^0 + PSite_{i,1}^1(t_1) \cdot \beta_2^1) \tag{3.10}$$

To reiterate what can be observed from (3.8); though similar to one this is not a linear model equation.

3.2 Optimization Techniques

Solving an optimization problem can be approached in a myriad of ways, though they can be split by their formulation specificities. The first and most prominent is single-objective-optimization (soo), optimizing for one equation often called penalty function, they have been around for centuries. A common method of finding a solution to a soo-problem is an iterative method like Newton's method, which calculate the derivative of the derivative of a function, i.e. the Hessian-matrix, to determine critical points. For complicated, constrained problems based on nonlinear equations Newton's method cannot be applied, instead quasi-Newton method find their use. One of such a method is sequential quadratic programming.

Second is multi-objective-optimization (moo), the objectives are two or more equations and can often not be solved using iterative methods. Instead a heuristic approach is chosen which determines the solution based not on its methematical feature but grounded on the ranking of proposed solutions acquired through a search algorithm. Such heuristics can also be applied on soo-problems. Genetic and evolutionary algorithms which improve by suggesting perturbed variations of the last, best proposition.

3.2.1 Sequential Least Squares Programming

The Problem defined above in Sec. 3.1 is as mentioned a nonlinear equation. Optimization problems with non linear equations (NLP), be it in the constraints or as the objective function, have the basic composition

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(\vec{x}) \tag{3.11a}$$

$$\text{subject to} \quad g_j(\vec{x}) = 0, \qquad j = 1, \ldots, m_e, \tag{3.11b}$$

$$g_j(\vec{x}) \geq 0, \qquad j = m_e + 1, \ldots, m, \tag{3.11c}$$

$$\vec{x}_j \leq \vec{x} \leq \vec{x}_u, \tag{3.11d}$$

where $f : \mathbb{R}^n \to \mathbb{R}^1$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be continuously differentiable and have no specific structure, both assumptions apply to the present Problem. Note that the bounds defined for α and β are not treated as inequality constraints, thus the section on handling inequality constraint is omitted, the quantity of equality constraints will nonetheless be denoted with m_e.

The basic algorithm for solving the problem (NLP) works through iterations, beginning with an initial vector of parameters $\vec{x}_0$ (or x0 in Sec. 4.2). Every following iteration $k+1$ of the iteration k results in a new parameter vector obtained by the step

$$\vec{x}_{k+1} := \vec{x}_k + \varepsilon_k d_k. \tag{3.12}$$

with d_k as the search direction of the iteration k and ε_k as its step length, or step size.

The search direction d_k is, since the utilized method is the `minimize(method='SLSQP')` function, procured by a linear least squares (LLS) subproblem. Proposed by Schittkowski in [33] and [34], with the quadratic approximation of the Lagrange function

$$\mathcal{L}(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \sum_{j}^{m} \lambda_j g_j(\vec{x}). \tag{3.13}$$

and the linear approximation of the constraints $g_j(\vec{x})$, the problem (LLS) has the baseline structure of

$$\begin{array}{ll} \underset{d \in \mathbb{R}^n}{\text{minimize}} & ||D_k^{\frac{1}{2}} L_k^T d + D_k^{-\frac{1}{2}} L_k^{-1} \nabla f(x_k)|| \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k) d = 0, \quad j = 1, \ldots, m_e. \end{array} \tag{3.14}$$

$$\underset{d \in \mathbb{R}^n}{\text{minimize}} \quad \left\| D_k^{\frac{1}{2}} L_k^T d + D_k^{-\frac{1}{2}} L_k^{-1} \nabla f(x_k) \right\| \tag{3.15a}$$

$$\text{subject to} \quad g_j(x_k) + \nabla g_j(x_k) d = 0, \quad j = 1, \ldots, m_e. \tag{3.15b}$$

The matrices D and L form the factorizations of the matrix B

$$B_k := L_k D_k (L_k)^T, \tag{3.16}$$

which in turn is the approximation matrix of the Hessian of the Lagrange function. $L = [l_1, \ldots, l_n]$ is a lower triangular matrix with unit diagonal entries and $D = diag(d_1, \ldots, d_n)$ is a diagonal matrix. This subproblem can be solved with the linear least squares algorithm of Lawson and Hanson [15], they use a modified Householder transformation to construct the diagonal matrices.

Step Size ε. Similar to Newton's method $\varepsilon = 1$ is the optimal value for general nonlinear functions near a local optimum, but the step size needs to be modified for vectors $\vec{x}_k$ far from the optimum, to guarantee global convergence.
In [13] it is proven that a one-dimensional minimization of the non-differentiable exact penalty function

$$\phi(x : \varrho) := f(\vec{x}) + \sum_{j}^{m_e} \varrho_j |g_j(\vec{x})| + \sum_{j=m_e+l}^{m} \varrho_j |g_j(\vec{x})|_- \tag{3.17}$$

with $|g_j(\vec{x})|_- := |min(0, g_j(\vec{x})|$ as a merit function $\varphi : R^l \rightarrow R^1$

$$\varphi(\alpha) := \phi(\vec{x}_k + \alpha d_k), \tag{3.18}$$

and $\vec{x}_k$ and d_k fixed, as well as knowledge of the upper bound for the Lagrange multipliers, leads to a step size ε guaranteeing global convergence for values of the penalty parameters ϱ_j greater than some lower bound. The penalty parameters ϱ are defined as:

$$\varrho_j := max(\frac{1}{2}(\varrho_j^- + |\mu_j|), |\mu_j|), \quad j = 1, \ldots, m, \tag{3.19}$$

where μ_j denotes the Lagrange multiplier of the constraint $g_j(\vec{x})$ in the least squares subproblem and ϱ_j^- is its penalty parameter of the previous iteration.
To overcome possible difficulties in the line search of the non-differentiable merit function (3.17), Schittkowski [33] substituted this by the differentiable augmented Lagrange function

$$\Phi(x,\lambda,\varrho) := f(x) - \sum_j^{m_e}(\lambda_j g_j(x) - \frac{1}{2}\varrho_j g_j(x)^2) \\ - \sum_j^{m} \begin{cases} \lambda_j g_j(x) - \frac{1}{2}\varrho_j g_j(x)^2, & \text{if } g_j(x) \le \frac{\lambda_j}{\varrho_j}, \\ \frac{1}{2}\frac{\lambda_j^2}{\varrho_j}, & \text{otherwise,} \end{cases} \tag{3.20}$$

Inconsistent Constraints. It might be possible that the constraints in subproblem (LLS) become inconsistent in the course of the iterations, although those of the original problem (NLP) are solvable. In order to overcome this difficulty, an additional variable δ is introduced into the least squares subproblem as in [29], leading to an (n+1)-dimensional subproblem with consistent constraints:

$$\text{minimize} \quad \left\| \begin{pmatrix} D^{\frac{1}{2}}L^T & :0 \\ 0 & :\rho \end{pmatrix} \begin{pmatrix} d \\ \delta_k \end{pmatrix} + \begin{pmatrix} D^{\frac{1}{2}}L^{-1}\nabla f(x) \\ 0 \end{pmatrix} \right\| \tag{3.21a}$$

$$\text{subject to} \quad (\nabla g_j(\vec{x}_k)d + \delta_k g_j(\vec{x}_k) = 0, \qquad j = 1,\ldots,m_e, \tag{3.21b}$$

and

$$0 \le \delta_k \le 1, \tag{3.22}$$

where $\delta_{j,k}$ has the value

$$\delta_{j,k} := \begin{cases} 1, & \text{if } g_j(\vec{x}_k) > 0 \\ \sigma_k, & \text{otherwise} \end{cases} \tag{3.23}$$

and it is made as large as possible, subject to the condition (3.22). The starting value for the augmented (LLS) can be the initial direction $(d_0,\delta_0)^T = (0,\ldots,0,1)^T$, since it satisfies the constraints above.

Update of the B-Matrix. The quasi-Newton method of updating the B-Matrix by approximation of the Hesse-Matrix of the Lagrange-function, facilitated with BFGS (more information on this strategy can be found in the paper [30]) was modified by [29] as seen below. These modifications keep the B-Matrix positive definite throughout the run when dealing with constrained problems.

$$B_{k+1} := B_k + \frac{q_k(q_k)^T}{(q_k)^T s_k} - \frac{B_k s_k (s_k)^T B_k}{(s_k)^T B_k s_k}, \tag{3.24}$$

with

$$\vec{s}_k := \vec{x}_{k+1} - \vec{x}_k = \varepsilon_k \vec{d}_k, \tag{3.25}$$

and

$$q_k := \theta_k \eta_k + (1-\theta_k) B_k \vec{s}_k, \tag{3.26}$$

where η_k is the difference in gradients of the Lagrange function

$$\eta_k := \nabla_{\vec{x}}\mathcal{L}(\vec{x}_{k+1}, \vec{\lambda}_k) - \nabla_x\mathcal{L}(\vec{x}_k, \vec{\lambda}_k), \tag{3.27}$$

and θ_k is chosen as

$$\theta_k := \begin{cases} 1, & \text{if } (\vec{s}_k)^T\eta_k \geq 0.2(\vec{s}_k)^T B_k\vec{s}_k, \\ \frac{0.8(\vec{s}_k)^T B_k\vec{s}_k}{(\vec{s}_k)^T B_k\vec{s}_k - (\vec{s}_k)^T\eta_k}, & \text{otherwise,} \end{cases} \tag{3.28}$$

which guarantees the condition

$$(\vec{s}_k)^T q_k \geq 0.2(\vec{s}_k)^T B_k\vec{s}_k, \tag{3.29}$$

The choice of conditions (3.25) - (3.28) guarantees the updated B-matrix (3.24) to remain positive definite for an arbitrary initial estimate of this matrix.

As mentioned above the (LLS) can be solved using an adapted Householder transformation. Schittkowski introduced further modifications in [33]. To summarise, the equality constraints in (3.21) are eliminated through Householder transformations. For the Hessian-matrix of the subsequent subproblem, instead of being approximated (B), its inverse is updated ($\bar{H}$). The new subproblem can then be reformulated as a least distance problem which provides the solution to the updating of the inverted Hessian-matrix. Same can be done using (3.24) as a starting point.

The updated B or H matrix can then be used to find the next best step in the function landscape. The penalty function above (3.19) provides the adequate step size and the direction.

Objective Definition

As shown SLSQP can only optimize for a single objective function, with the capability of handling highly nonlinear constraints. To accommodate for this restriction two error vectors are introduced.

$$\vec{f}_{i,-} = \{f_{i,-}(t_1), \dots, f_{i,-}(T)\} \tag{3.30}$$
$$\vec{f}_{i,+} = \{f_{i,+}(t_1), \dots, f_{i,-}(T)\} \tag{3.31}$$
$$\tag{3.32}$$

The $\vec{f}_{i,-}$ vector is the amount necessary to subtract from the estimation to equal the measurement of mRNA i, whereas $\vec{f}_{i,+}$ is the amount added to the estimation. $\vec{f}_{i,-}$ and $\vec{f}_{i,+}$ could theoretically take values between zero and infinity, but are, to reduce the search space, constraint between zero and 20. The optimization algorithm tries to minimize these two vectors. The constraints are the equation (3.8) for each mRNA, the α- and

β-constraints in (3.7a) to (3.7d) and the restriction for the error vectors.

$$\begin{aligned}
&\underset{X}{\text{minimize}} && \sum_{mRNA} \vec{f_{i,-}} + \vec{f_{i,+}} && \text{(3.33a)}\\
&\text{subject to} && = \vec{R}_i - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i + \vec{f_{i,-}} - \vec{f_{i,+}} && \text{(3.33b)}\\
& && 1 = \sum_{j=1}^{m} \alpha_{i,j} && \text{(3.33c)}\\
& && 1 = \sum_{q=0}^{k} \beta_j^q && \text{(3.33d)}\\
& && 0 \leq f_-, f_+ \leq 20 && \text{(3.33e)}\\
& && 0 \leq \alpha_{i,j} \leq 1 && \text{(3.33f)}\\
& && -2 \leq \beta_j^q \leq 2. && \text{(3.33g)}
\end{aligned}$$

In the approach above every time point is associated with one negative error and one positive error, meaning for n mRNA in a network with each having T time points, $2nT$ error values are necessary for this equation. To reduce the complexity of the model for networks with a high amount of genes a second version is constructed using, instead of the absolute difference between the estimation and measurement, a root mean square deviation. Thus only $2n$ error values are needed. All other constraints are as before.

$$\begin{aligned}
&\underset{X}{\text{minimize}} && \sum_{mRNA} \vec{f_{i,-}} + \vec{f_{i,+}} && \text{(3.34a)}\\
&\text{subject to} && 0 = \sqrt{\frac{\sum_t (\vec{R}_i - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i)^2}{T}} + \vec{f_{i,-}} - \vec{f_{i,+}} && \text{(3.34b)}\\
& && 1 = \sum_{j=1}^{m} \alpha_{i,j} && \text{(3.34c)}\\
& && 1 = \sum_{q=0}^{k} \beta_j^q && \text{(3.34d)}\\
& && 0 \leq f_-, f_+ \leq 20 && \text{(3.34e)}\\
& && 0 \leq \alpha_{i,j} \leq 1 && \text{(3.34f)}\\
& && -2 \leq \beta_j^q \leq 2. && \text{(3.34g)}
\end{aligned}$$

3.2.2 Non-dominated Sorting Genetic Algorithm 2

The multi-objective evolutionary algorithm (MOEA), the second version of non-dominant sorting genetic algorithm, abbreviated with NSGA-2, implemented in the python package `pymoo` [3], based on DEBs paper [6], will be explained below. Essentially it uses a

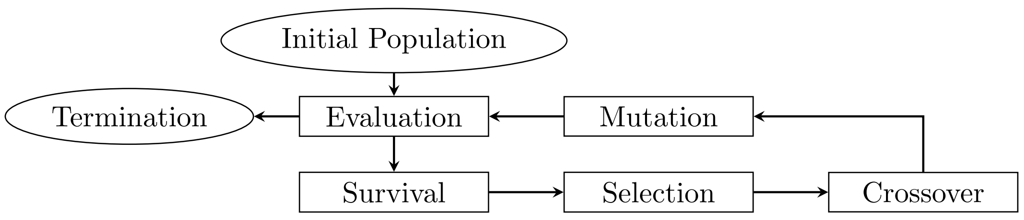

Figure 3.1: General outline of a genetic algorithm.

domination rank, equal to a solutions pareto-front level, and a crowding distance, which measures the density of solutions around another, to determine the best set of parameters in a given generation.
In Fig. 3.1 the general outline of a genetic algorithm is depicted. This will serve as an orientation for the structure and progression of the NSGA-2. In the following section the nodes will be explained based on [6], [7] and [21].

Initialization. The first step for NSGA-2 is the initialization of a starting population. This population is acquired using the FloatRandomSampling (FRS) function from `pymoo`, which applies the the `numpy` function `random.random` to sample between the given bounds of the Problem. The `numpy` function is based on the Mersenne Twister [23], which will not be explained, it shall suffice to say it is a close approximation to randomness. This FRS results in NP vectors of size D, where NP is the size of the population and D the dimension, i.e. the number of parameters.

Evaluation. For the evaluation of the solutions in each generation the objective functions are calculated and checked for constraint violations (CV), should the termination criterion be fulfilled the algorithm ends, otherwise it moves on to **Survival**. The termination criterion for this application is a function tolerance. Should the objective function values change by less than a user specified value, here 0.005, the algorithm terminates.

Survival. To acquire the aforementioned domination rank a definition for 'domination' is required.
Definition 1: A solution z_1 dominates a solution z_2 if

- no result of z_2 is less than the result of z_1

  AND

- at least one of the results of z_2 is strictly greater than that of z_1

If a solution z is not dominated by any other it is called **non-dominated** [37].
Additionally to this Definition a second one is necessary, since the effect of constraints are not yet accounted for.
Definition 2: A solution z_1 is said to constrained-dominate a solution z_2 if any of the following conditions are true:

1) solution z_1 is feasible, solution z_2 is not,

2) solutions z_1 and z_2 are both infeasible, but z_1 has a smaller overall CV,

3) solution z_1 and z_2 are both feasible, but solution z_1 dominates z_2.

As before, a solution z is non-constrained-dominated when no other solution constrained-dominates solution z.
The definition of constrained-domination will now serve as the default domination.
By comparing every solution to all other solutions in a generation one can acquire two entities, n_z and $\mathcal{S}_z$. The value n_z represents the number of solutions dominating solution z, the domination count. $\mathcal{S}_z$ is the set of solutions dominated by solution z. After iterating through the population any solution with a domination count of zero is put in a separate list, the first non-domination front, or first pareto-front, $\mathcal{D}_0$. The subsequent pareto-front is determined by reducing the domination count n_q of solution q in $\mathcal{S}_z$ by one, repeating for every solution z in $\mathcal{D}_0$; all solutions with a now n_q of zero are put in the next list, the second domination front $\mathcal{D}_1$. This scheme is repeated until all fronts are identified and thus the solutions in a population are sorted.

Preserving the diversity of a population is necessary for both fast convergence and determining the true global minimum, therefore the crowding distance is introduced. This value represents the density of solutions around a specific solution z.

To calculate the crowding distance (cd) the pareto-front needs to be sorted by the function value of each objective, afterwards the extreme solutions are assigned a cd of infinity. The solutions in between, get their cd by first calculating the distance between the two adjacent solutions, dividing these distances by the norm, summing along the objective axis. To get a comparable distance for every solution the sum needs to be divided by the objective number.
With this a Crowded-Comparison Operator can be introduced. This operator requires the non-domination rank, the domination-front level of a solution, and the crowding distance of a solution. A partial order $\prec_n$ is defined as $z_1 \prec_n z_2$ if

- $(\mathcal{D}(z_1) < \mathcal{D}(z_2))$

  OR

- $((\mathcal{D}(z_1) = \mathcal{D}(z_2))$ AND $(cd_{z_1} > cd_{z_2}))$.

Thus solution z_1 is preferred if it has a lower non-domination rank, i.e. low pareto-front level, or, should both be equal, a higher crowding distance.

Now with both these tools, non-dominated sorting and crowded-comparison operator, we can look at how a population, with offspring population, finds the best solution in said population. The combined population at iteration k will be denoted with R_k. This R_k is non-dominated sorted until the solutions in the determined fronts equal or exceed NP, all other solutions are rejected. Should a front extend beyond NP, using the crowed-comparison operator the best solutions, i.e. higher cd, are assessed and kept. This process is depicted in the Fig. 3.2 below.

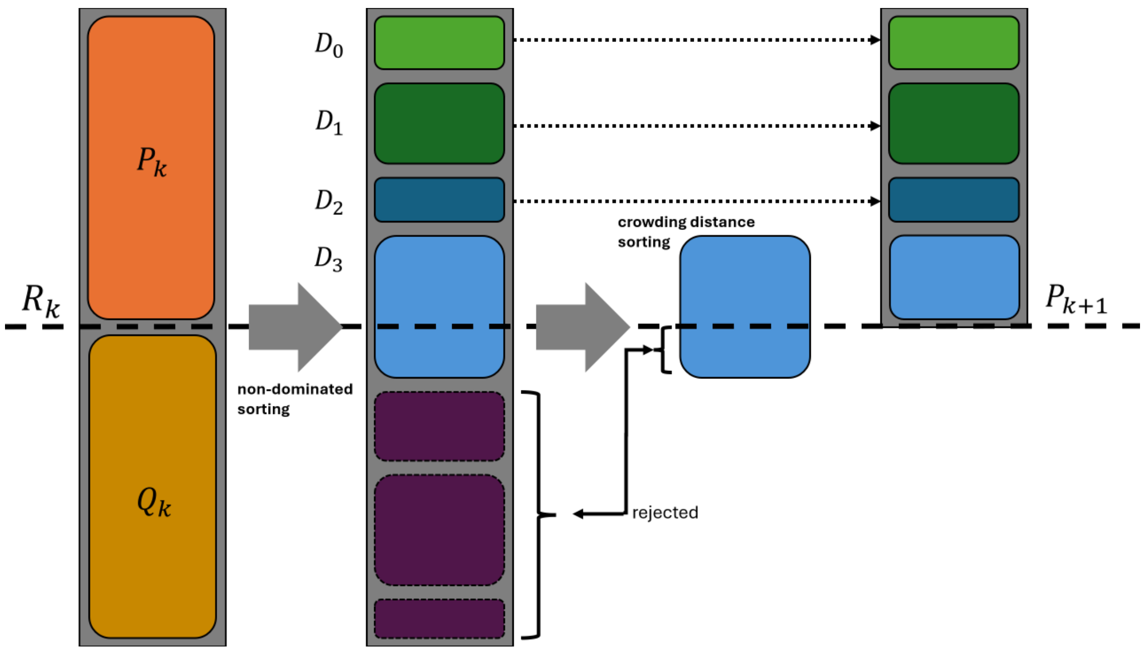

Figure 3.2: The process of the NSGA-2 survival. The parent population P_k and offspring population Q_k are both of size NP, the dashed line marking size NP, and form the population R_k in the first column. The second column has sorted domination fronts, all those which cannot be accommodated for in NP are automatically rejected. The front $\mathcal{D}_3$ exceeds the size NP and is reduced with the corwded-comparison operator. The process ends with a new parent population P_{k+1} of size NP.

Selection. The pymoo implementation of the NSGA-2 uses a series of decision points for a binary tournament to determine possible mating partners. First the domination is checked, if there are no CVs they move on to the next point; should both have equal, non zero CVs a random solution is chosen. The next step examines who dominates whom and selects the dominator as winner of the tournament. If both solutions so far are equal, their *cd* is calculated, the higher value dictates the winner. This results in a set of parent solutions for the next section.

Crossover. This is the mating process of NSGA-2, utilizing the simulated binary crossover (SBX) from [7]. Calculating the offsprings from two parent solutions requires the spread factor ψ_i, defined as the ratio of the absolute difference of the offspring values and the absolute difference of the parent values.

$$\psi_i = \left| \frac{z_i^{(2,k+1)} - z_i^{(1,k+1)}}{z_i^{(2,k)} - z_i^{(2,k)}} \right| \tag{3.35}$$

This unknown spread ψ_i factor is determined by solving the probability distribution function (3.36) below. This is enabled by the randomly chosen value $u_i \in [0,1]$, designated as the area under the probability function, its integral, from zero to ψ_i.

$$\mathcal{P}(\psi_i) = \begin{cases} 0.5(\eta_c+1)\psi_i^{\eta_c}, & \text{if } \psi_i \leq 1 \\ 0.5(\eta_c+1)\psi_i^{\frac{1}{\eta_c+2}} & \text{otherwise.} \end{cases} \tag{3.36}$$

The value η_c is the distribution index, a non-negative real number, dictating the probability for creating an offspring near or distant to the parent, set by the user. A low η_c flattens the curve, increasing the probability for a distant offspring, whereas a high value results in a higher probability for near offsprings. These offsprings are, after the computation of ψ_i, calculated using the equations (3.37) and (3.38) below.

$$z_i^{(1,k+1)} = 0.5\left[(1+\psi_i)\cdot z_i^{(1,k)} + (1-\psi_i)\cdot z_i^{(2,k)}\right] \tag{3.37}$$

$$z_i^{(2,k+1)} = 0.5\left[(1-\psi_i)\cdot z_i^{(1,k)} + (1+\psi_i)\cdot z_i^{(2,k)}\right] \tag{3.38}$$

SBX results after application to all variables of the parent solutions in two new offsprings, doubling the population size from NP to 2NP.

Mutation. All solutions in a generation have a probability p_m to be mutated as the last step in their cycle. This mutation is facilitated through polynomial mutation [21]. Every value of a solution z has the probability p_m of being mutated, should a randomly drawn number $R_i \in [0,1]$ be below or equal to this probability, the polynomial mutation is executed. Acquiring the mutation factor δ_m is ruled by another randomly drawn number $r \in [0,1]$. This r specifies the sampling space, lying either between z_i^l and z_i or between z_i and z_i^u, therefore sampling on the lower bound side or the upper bound side. A decision is made as follows

$$\delta = \begin{cases} \frac{z_i - z_i^l}{z_i^u - z_i^l}, & \text{if } r \leq 0.5 \\ \frac{z_i^u - z_i}{z_i^u - z_i^l}, & \text{otherwise.} \end{cases} \tag{3.39}$$

And for δ_m

$$\delta_m = \begin{cases} \left[2\cdot r + (1-2\cdot r)(1-\delta)^{\eta_m+1}\right], & \text{if } r \leq 0.5 \\ 1 - \left[2(1-r) + 2(r-0.5)(1-\delta)^{\eta_m+1}\right]^{\frac{1}{\eta_m+1}}, & \text{otherwise.} \end{cases} \tag{3.40}$$

Thereafter the mutated value is calculated using (3.41).

$$z_m = z_p + \delta_m(z_i^u - z_i^l) \tag{3.41}$$

With this the NSGA-2 cycle is completed, the population would move on to the evaluation and continue until the termination requirements are fulfilled.

Objective Definition

The NSGA-2 algorithm is capable to optimize for multiple objectives, hence the optimization definition slightly deviates. Instead of optimizing for an error vector the objectives are the differences, or root mean square deviations, directly. All other constraints stay the same. The target definition is hence written as below for the single version (NSGA2_single), i.e. calculating for all time points.

$$\begin{aligned}
\underset{\alpha,\beta}{\text{minimize}} \quad & \vec{R}_i - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i && (3.42a)\\
\text{subject to} \quad & 1 = \sum_{j=1}^{m} \alpha_{i,j} && (3.42b)\\
& 1 = \sum_{q=0}^{k} \beta_j^q && (3.42c)\\
& 0 \le f_-, f_+ \le 20 && (3.42d)\\
& 0 \le \alpha_{i,j} \le 1 && (3.42e)\\
& -2 \le \beta_j^q \le 2. && (3.42f)
\end{aligned}$$

For the *rmsd* version (NSGA2_rmsd) the following problem is optimized:

$$\begin{aligned}
\underset{\alpha,\beta}{\text{minimize}} \quad & \sqrt{\frac{\sum_t (\vec{R}_i - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i)^2}{9}} && (3.43a)\\
\text{subject to} \quad & 1 = \sum_{j=1}^{m} \alpha_{i,j} && (3.43b)\\
& 1 = \sum_{q=0}^{k} \beta_j^q && (3.43c)\\
& 0 \le f_-, f_+ \le 20 && (3.43d)\\
& 0 \le \alpha_{i,j} \le 1 && (3.43e)\\
& -2 \le \beta_j^q \le 2. && (3.43f)
\end{aligned}$$

3.2.3 Differential Evolution

The single objective evolutionary algorithm differential evolution (DE) has a similar structure to a genetic algorithm, the structure below in Fig. 3.3 reveals the deviations from NSGA-2. In essence the algorithm adds a weighted difference to a the best vector of the population, performs a crossover between this vector and a target vector, select either offspring or target solution by a greedy criterion, mutates the chosen vector and repeats until the termination requirement is satisfied. The algorithm can be modified to an extent, the discussed version is `DE/best/1/bin`.[31]

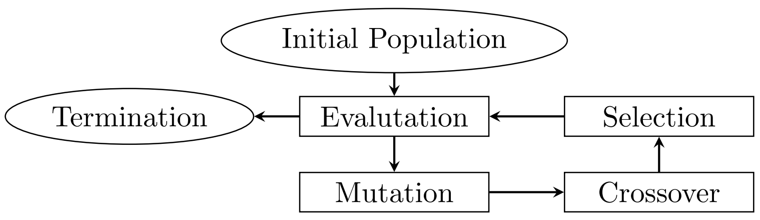

Figure 3.3: General outline of the evolutionary algorithm differential evolution.

Initialization. The population for the DE is sampled with the LatinHypercube-Sampler (LHS). Similar to the FRS used for NSGA-2, the underlying process of LHS tries to achieve near random results, further description can be read in [22] or in [24]. The LHS samples for NP vectors of size D, yielding an adequate coverage of the parameter space. Is a known solution or similar available, the DE initializes the population by adding normally distributed random deviations to this solution.

Evaluation. For the evaluation the objective function the algorithm uses the cost function and assess the available solutions for their constraint violations. If the termination criterion is fulfilled the algorithm ends.

Mutation. To apply a mutation to a solution $z_{best,G}$, in this variation the best, two additional solutions are involved, $z_{r_1,G}$ and $z_{r_2,G}$. These vectors are chosen at random from the population, $r_1, r_2 \in \{1, 2, \ldots, NP\}$, whereby $r_1 \neq r_2 \neq r_3$ needs to be true. The new mutated vector $\nu_{i,G}$ is computed with (3.44)

$$\nu_{i,G+1} = z_{r_1,G} + F_W \cdot (z_{r_2,G} - z_{r_3,G}) \tag{3.44}$$

Where F_W refers to the weight factor of the mutation, amplifying the differential variation. It is usually chosen from [0,2].

Crossover. The crossover is applied to a fourth vector $z_{i,G}$ with the crossover partner $\nu_{i,G+1}$. The decision which value to chose is made by generating two random numbers, $randb(j) \in [0, 1]$ and $rnbr(i) \in 1, 2, \ldots, D$.

$$c_{i,j,G+1} = \begin{cases} \nu_{i,j,G+1}, & \text{if } ((randb(j) \leq CR) \ \text{ OR } \ j = rnbr(i) \\ z_{i,j,G}, & \text{if } ((randb(j) > CR) \ \text{ AND } \ j \neq rnbr(i) \end{cases} \tag{3.45}$$

This ensures at least one crossover between the target solution and the mutated vector.

Selection. The offspring from the crossover is then compared to the target solution. The survival/selection function chooses, in case of present constraints, the offspring solution $c_{i,G+1}$ if any of the following statements are true.

- $z_{i,G}$ and $c_{i,G+1}$ are both infeasible, but the constraints improved
- one result became feasible

- $z_{i,G}$ and $c_{i,G+1}$ are both feasible, but the cost function value improved.

With this the algorithm proceeds on to the evaluation of the proposed solutions and terminates, if any of the terminations requirements are fulfilled, or moves to the next mutation.

Objective Definition

Similar to the SLSQP method DE can only handle a single objective function, therefore the same adjustments have to be introduced to accommodate for this restriction.

$$\vec{f_{i,-}} = \{f_{i,-}(t_1), \ldots, f_{i,-}(T)\} \tag{3.46}$$

$$\vec{f_{i,+}} = \{f_{i,+}(t_1), \ldots, f_{i,-}(T)\} \tag{3.47}$$

The $\vec{f_{i,-}}$ vector is again the amount necessary to subtract from the estimation to equal the measurement of the mRNA i, and $\vec{f_{i,+}}$ is the amount added to the estimation. The bounds for $\vec{f_{i,-}}$ and $\vec{f_{i,+}}$ are reduce to be between zero and 20, same as done for SLSQP. The constraints are the equation (3.8) for each mRNA, the α- and β-constraints in (3.7a) to (3.7d) and the restriction for the error vectors.

$$\underset{X}{\text{minimize}} \quad \sum_{mRNA} \vec{f_{i,-}} + \vec{f_{i,+}} \tag{3.48a}$$

$$\text{subject to} \quad = \vec{R_i} - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta_j} \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha_i} + \vec{f_{i,-}} - \vec{f_{i,+}} \tag{3.48b}$$

$$1 = \sum_{j=1}^{m} \alpha_{i,j} \tag{3.48c}$$

$$1 = \sum_{q=0}^{k} \beta_j^q \tag{3.48d}$$

$$0 \leq f_-, f_+ \leq 20 \tag{3.48e}$$

$$0 \leq \alpha_{i,j} \leq 1 \tag{3.48f}$$

$$-2 \leq \beta_j^q \leq 2. \tag{3.48g}$$

The same attempt is made for the DE algorithm as for all others, replacing the single time point orientated optimization (DE_single) approach by a root mean square deviation (DE_rmsd).

$$
\begin{aligned}
\underset{X}{\text{minimize}} \quad & \sum_{mRNA} \vec{f_{i,-}} + \vec{f_{i,+}} && \text{(3.49a)} \\
\text{subject to} \quad & 0 = \sqrt{\frac{\sum_t (\vec{R}_i - \begin{bmatrix} \mathbf{P}_j \cdot \vec{\beta}_j \cdot \vec{TF}_j \\ \vdots \\ \mathbf{P}_m \cdot \vec{\beta}_m \cdot \vec{TF}_m \end{bmatrix} \cdot \vec{\alpha}_i)^2}{T}} + \vec{f_{i,-}} - \vec{f_{i,+}} && \text{(3.49b)} \\
& 1 = \sum_{j=1}^{m} \alpha_{i,j} && \text{(3.49c)} \\
& 1 = \sum_{q=0}^{k} \beta_j^q && \text{(3.49d)} \\
& 0 \leq f_-, f_+ \leq 20 && \text{(3.49e)} \\
& 0 \leq \alpha_{i,j} \leq 1 && \text{(3.49f)} \\
& -2 \leq \beta_j^q \leq 2. && \text{(3.49g)}
\end{aligned}
$$

3.3 Metrics for Evaluation

As metrics for the evaluation of the methods applied on the Problem the simplistic the categories 'Time', 'Constraint-Violation' and 'Mean-Error' are considered.

'Time' takes the time it took for the algorithm to either find an optimal solution or terminate of other reasons into account.

'Constraint-Violation' looks at all defined constraints and how an algorithm handled them. A solution is excluded from further analysis when constraints are substantially unsatisfied.

'Mean-Error' includes two error calculations, mean absolute error (MAE) and root mean squared deviation (RMSD), between the measurements and the estimations.

Additionally the category 'Physicality' looks at the parameter estimations and if the solutions come to congruent results among each other and with available experimental data.

Solutions which transgress the constraints or cannot to imitate the progression of mRNAs fail at simulating the network, hence they not are not considered any further than their respective deficiency.