Chapter 7

Optimisation

The possibility of fast and practically arbitrary laser pulse modification with pulse shaping techniques described in Sec. 5.6 provides a very effective tool for an active control of molecular processes. This control can be obtained through the interaction of shaped laser pulses with a molecular system, where a closed loop between the pulse shaper and the resulting response of the molecular system is utilised together with a self-learning algorithm [JRa92]. Such scheme of an adaptive closed loop allows one to control different photoinduced processes such as bond dissociation and rearrangement [LMR01], selective fragmentation [ABB98], laser induced fluorescence [BYW97], or high harmonic generation [BBZ00].

This chapter presents concepts of adaptive closed loop optimisation. It begins with a general overview of excising optimisation methods. Then methods based on stochastic search are described in detail. Finally, the practical implementation of the optimisation algorithm employed in the present work is described.

7.1 Introduction to Mathematical Optimisation

The aim of an optimisation is to maximise or minimise some quantity by systematically modifying the values of parameters within an allowed range of variation. The quantity which has to be optimised is called an objective or cost function. A set of parameters (unknowns or variables) constitutes the optimisation variable. An ensemble of optimisation variables constitutes the search space. The restrictions on allowed parameter values are known as "constraints". The constraints allow the unknowns to have only certain values and exclude others. Generally speaking, constraints are not essential.

Mathematically, the optimisation problem is formulated as

$$\text{maximise } f(x), \text{ subject to } c_i(x) \leq b_i, \ i = 1, ..., m \quad , \tag{7.1}$$

where the vector $x = (x_1, ..., x_n) \in \mathbf{R}^n$ is the optimisation vector of n independent variables, $f(x) : \mathbf{R}^n \rightarrow \mathbf{R}$ is the objective or cost function, $c_i(x) : \mathbf{R}^n \rightarrow \mathbf{R}$ are the constraint functions, and b_i are the bounding conditions. The constraints in the form of $c_i(x) = b_i$ are termed equality constraint functions, while those in the form of $c_i(x) < b_i$ are inequality constraint functions. Generally, a minimisation of a function $f(x)$ can be regarded as a maximisation of $-f(x)$. The purpose of an optimisation is to find the global maximum – i.e. a vector x_G that maximises the objective function $f(x)$ over all possible vectors x

$$f(x_G) > f(x) \ \ \forall x \in D(x), \ \ x \neq x_G \quad , \tag{7.2}$$

where $D(x)$ is set of feasible values of the vector x. Obviously, for an unconstrained problem $D(x)$ is infinitely large.

In the general case several local maxima can exist in the search space. A local maximum is defined as a vector x_L which satisfies the following condition

$$f(x_L) > f(x) \ \ \forall x \in N(x_L, \delta), \ \ x \neq x_L \quad , \tag{7.3}$$

where $N(x_L, \delta)$ is defined as the set of vectors x contained in the neighbourhood of x_L (within some arbitrarily small distance δ of x_L).

A large variety of mathematical methods was developed to solve optimisation problems. Many methods are suitable only for certain types of problems. Therefore, it is important to recognise the problem type and find an appropriate solution technique. Complexity of an optimisation problem depends on relationships between the objective and constraint functions. The simplest class comprises linear optimisation problems. An optimisation problem is linear if the objective and all constraints are linear functions of the decision variables. If the objective or at least one of the constraints is a smooth nonlinear function, the corresponding optimisation problem is called a smooth nonlinear optimisation problem. The most general class of optimisation problems is a non-smooth optimisation problem.

Linear optimisation problems can be solved using the simplex method [CLR01]. Alternatively, the interior point method (also referred to as barrier method) was developed to solve this type of optimisation problem [Wri97]. There is no single method which is the best for all

smooth nonlinear optimisation problems. The most widely used and effective methods are the gradient projection and reduced gradient methods [WXi00]. Non-smooth optimisation problems can have multiple locally optimal points. Therefore, since gradient information can not be used to determine the direction in which the function is increasing (or decreasing), methods based on either the systematic or random search are preferable for the solution of such type of optimisation problems. The classical group of methods based on the systematic search is called branch and bound methods [Moo91]. These methods are methods for global optimisation. Their efficiency depends critically on the effectiveness of the branching and bounding algorithms used. There is no universal bounding algorithm that works for all problems. Wrong choices can lead to a slowdown of the method. In the worst case the computation time grows exponentially with the problem size. Random search methods are non-deterministic and stochastic. Therefore, they can provide different solutions on different runs, even when starting from the same point on the same model. Generally, these methods are not able to prove that the solution obtained is the optimal solution. However, these stochastic optimisation methods are very fast. The application area for these methods comprises tasks, where systematic search methods fail. Many problems of practical interest fall into this category, for example, those where the search space is too large or the underlying mathematical model is not known exactly. The next section describes several examples of optimisation methods from this group in detail.

7.2 Methods of Stochastic Optimisation

7.2.1 Simulated Annealing

A very popular stochastic algorithm which can be used efficiently for a wide range of optimisation problems is simulated annealing (SA) [KGV83]. The original motivation of this method comes from the statistical physics by an analogy with the cooling of fluids into a crystalline structure of minimal energy.

The SA is just an iterative improvement [DFF63] incorporating with the Metropolis criterion [MRR53] for accepting or rejecting of a randomly generated trial move. The main advantages of SA over simple iterative improvement lies in the ability to avoid trapping into locally optimal

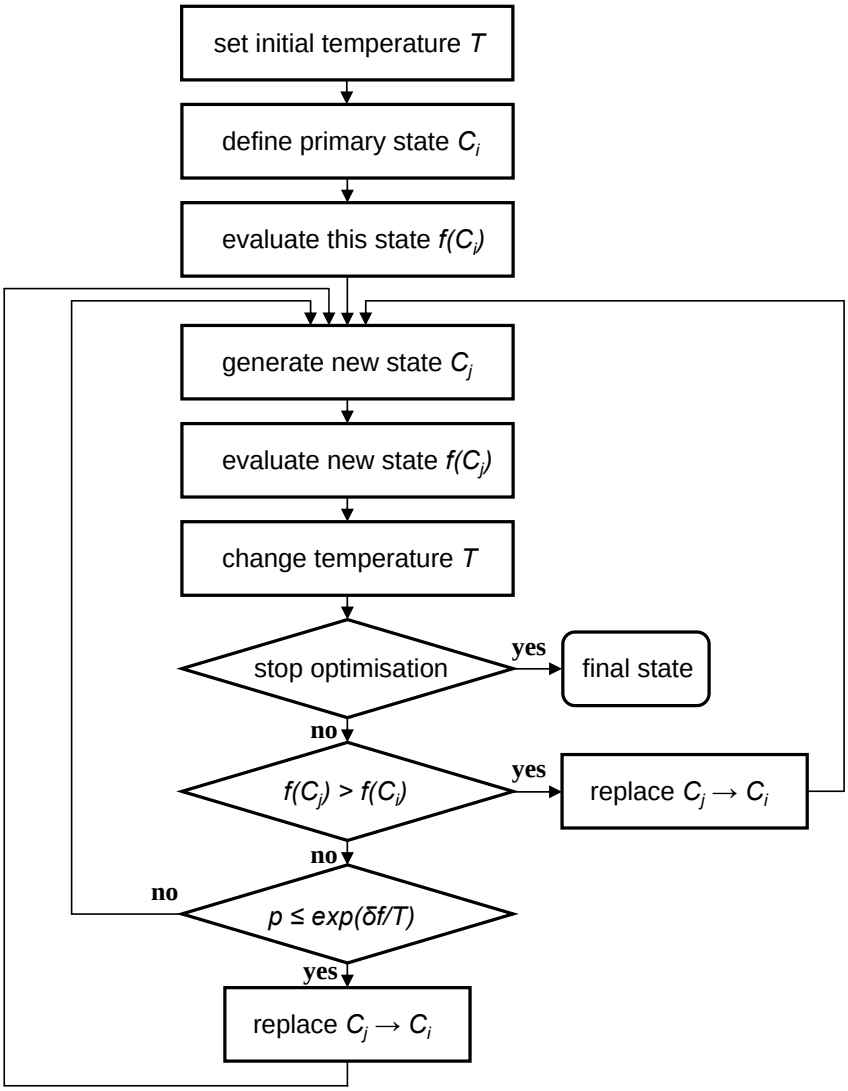

Figure 7.1: Diagram of the simulated annealing algorithm. For details see text.

points.

The structure of the SA algorithm for maximisation is depicted in Fig. 7.1. It starts with setting an initial system temperature T and the definition of a primary (initial) system configuration C_i. It is important to mention that the temperature here is not a real temperature but a control parameter which is called temperature by analogy with statistical physics. Then the primary system configuration is evaluated resulting in a number of a scalar objective function $f(C_i)$. After that, a new system configuration C_j is generated by random changes of the old configuration. The changes have to be rather small, but at the same time they must allow to

reach any possible configuration of the system within a finite number of iterations. Then the new configuration is evaluated by calculating $f(C_j)$. After that, the temperature of the system is changed. Now, if the value of the objective function for the new system configuration is larger than for the primary configuration ($f(C_j) > f(C_i)$), the old configuration is replaced by the new one ($C_j \rightarrow C_i$). Otherwise the replacement can occur with some probability p depending on the temperature

$$p = \exp\left(\frac{\delta f}{T}\right) \quad , \tag{7.4}$$

where $\delta f = f(C_j) - f(C_i)$ is the increase of the objective function and T is a current temperature. This "temperature" is a control parameter measured in the same units as the objective function. The procedure is repeated in the loop until stopping conditions or a final temperature is reached. The SA algorithm for minimisation has only insignificant differences. Namely, the old configuration is replaced by the new one ($C_j \rightarrow C_i$) if $f(C_j) < f(C_i)$, otherwise the replacement probability is given by Eq. (7.4) with $\delta f = f(C_i) - f(C_j)$. For fast and successful performance of the algorithm, it is very important to judiciously set the initial temperature and the rules for reducing it. An appropriate initial temperature can be found using the following procedure [Kir84]. The SA algorithm is started a few hundred times with some arbitrary temperature to determine a fraction of accepted system configurations according to Eq. (7.4). If this fraction is less than $80\,\%$, the initial temperature has to be doubled. This process is repeated until the fraction overcomes this threshold. During optimisation the temperature can be decreased either in an exponential cooling scheme

$$T_j = \alpha T_i \tag{7.5}$$

with $\alpha < 1$ or in a linear cooling scheme

$$T_j = T_i - \Delta T \tag{7.6}$$

with a small ΔT. The final temperature is either predetermined by setting the total number of iterations or, alternatively, the optimisation can be interrupted if no further progress is observed.

SA has advantages and disadvantages compared to other global optimisation techniques. Among its advantages are the relative ease of implementation and the ability to provide reasonably good solutions for many combinatorial problems. Its drawbacks include the need for a great deal of computer time for many runs and carefully chosen tunable parameters [ECF98].

7.2.2 Evolutionary Algorithms

The next group of optimisation algorithms (genetic algorithm [Hol73], evolution strategy [Rec73], and genetic programming [Koz92]) attempts to simulate principles of biological evolution processes. These are so-called evolutionary algorithms.

The genetic algorithm (GA) is based on the phenomena of natural evolution and survival of the fittest. Its terminology is drawn from the evolution theory of Darwin [Dar01]. The main difference between GA and SA is that the SA works with one solution, while the GA operates with a set of solutions (a population of individuals in the evolution terminology). The process of transforming one population to another is described by the term "generation" or "iteration". The construction of a new generation involves specific operators of selection, crossover, and mutation. Usually, an objective function for maximisation with GA is referred to as "fitness function" to accentuate the evolutionary nature of this algorithm.

The basic structure of the GA algorithm is presented in Fig. 7.2. The optimisation begins with building of an initial population of N individuals. This can be done either by a random generator or by some problem specified guess if possible. The GA employs a binary representation of individuals, where each individual is encoded by a bit string (chromosome). The chromosome length determines the accuracy with which an individual can be encoded.

Then all individuals are evaluated by applying the fitness function. This function returns a scalar number (fitness) depending on the fitness of the individual which is taken as the function input. Assuming that the aim of the optimisation is maximisation, higher values of the fitness function correspond to more successful individuals.

After this evaluation some individuals have to be selected according to their fitness and used for reproduction to create a new generation of individuals. There are several methods of selection. All methods imply that the individuals with a higher fitness have a larger probability to be chosen for the reproduction. One of the commonly used and most simple among them is the roulette wheel selection (proportional selection) method [Bak87]. The probability of any individual to be selected for reproduction in this method is proportional to its fitness value. Unfortunately, the roulette wheel selection method is affected by a scaling problem when the selection probabilities become strongly dependent on the scaling of the fitness function [BHo91]. Another popular selection method is called tournament selection [GDe91]. Here the best individual according to the fitness chosen from two randomly taken individuals is used

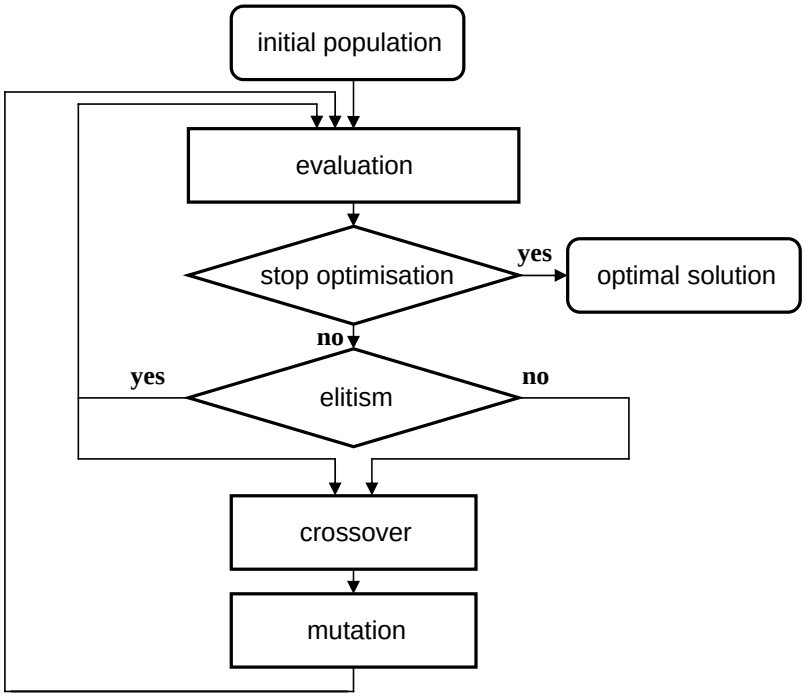

Figure 7.2: Basic scheme of the genetic algorithm. For details see text.

for reproduction. This method can be generalised involving a larger number of the individuals to pick up the best one. The selection method, when some fraction of the individuals with the highest fitness are selected and then these individuals utilised for the reproduction with the same probability, is known as the "truncation selection" [BTh95]. Usually, the truncation threshold is set to select between 10% and 50% of the population for the reproduction. For the ranking selection the individuals are ranked according to their fitness values. The highest rank corresponds to the best individual. Then the individuals are selected for the reproduction with the probability proportional to the their rank. The case of the probability linearly assigned to the rank corresponds to the linear ranking selection [BTh95]. For the exponential ranking selection [BTh95] the probability has to be exponentially weighted with the rank. Sometimes an individual is characterised not only with a single value of the fitness function, but additional criteria are involved in order to evaluate the quality of the individual. Then, the "multi-objective fitness assignment" has to be applied [Fon95].

A new generation of individuals is created by crossover and mutation using individuals selected with one of the above described methods. In the simplest form of the one point

crossover a chromosome part of one individual is exchanged with a chromosome part of another individual producing an offspring. The split point of both chromosomes is defined randomly. Generally, up to $n-1$ points crossover can be performed (n is a chromosome length) but the optimisation performance is degraded when many points are employed for the crossover [Jon75]. The idea of the multi-point crossover is that parts of the chromosome that contribute most to the fitness of a particular individual may not necessarily be contained in adjacent substrings [Boo87]. The next type of crossover is the so-called "uniform crossover" operator which creates offsprings by picking each bit from either of the two parent chromosomes [Sys89]. The uniform crossover operator produces on the average $n/2$ crossings of a chromosome. The "shuffle crossover" is similar to the uniform crossover [CES89]. The variables are randomly shuffled in both parents before the exchange. After one point crossover, the variables in the offsprings are unshuffled in reverse. Then new individuals created by the crossover undergo the mutation. Each bit of the chromosome can be flipped from 0 to 1 or from 1 to 0 with some probability. Since the initial population of individuals may not contain enough variability to reach the optimal solution via the crossover alone, the mutation is used to introduce extra variability. There are several modifications of the GA with respect to the crossover and the mutation. For example, the crossover can be applied with some probability and then the mutation takes place only for individuals for which the crossover fails. Sometimes a number of possible mutations is restricted to only one per an individual. Commonly, a probability of the mutation is very low, while the crossover is applied with a high probability.

If all individuals of a new generation are created by crossover and mutation, such process is called the "total replacement". Every individual lives one generation only. But it can lead to the situation when the best individuals are replaced with worse offspring, and therefore good information is lost. To avoid this, a number N_e of the best individuals can be transferred into a new generation without any modifications. This is so-called "elitism replacement". The rest $N-N_e$ individuals are created by the crossover and the mutation. Assuredly, these N_e of the best individuals in the original population are also available for the crossover and the mutation. The elitism guarantees that the best chromosomes will be always presented in succeeding generations. The proper rate of the individuals N_e/N involved in the elitism affects the success of an optimisation. If the output of the fitness function is noise free, it is enough to send only one individual to the next generation. But, if the fitness function contains also

an experimental noise, at least several individuals have to be used to be sure that the best individual is selected among them. A too large rate can create a predominance of particular type of chromosomes in a population. This reduces the diversity of individuals required for an adequate search and even can be a reason of the premature convergence to a local optimum. The high selection probability exclusively given during the selection procedure to the individuals with large fitness values especially at the initial stage of the optimisation process can again be a reason of the premature convergence to a local optimum. Sometimes, the "fitness based replacement" is applied. In this case the new generation is built from individuals with the highest fitness taken from both offspring and the old generation.

Then a new generation is evaluated again and the optimisation loop is run until stop conditions, such as a preselected number of generations or some desired fitness value, are reached. Alternatively, a stop point can be determined according to the evaluation of the fitness function. For example, if the difference between the best fitness value and the average fitness value over the same generation is rather small.

The "evolution strategy" (ES) is similar to GA [Rec73]. In the general case λ offspring individuals are created from μ parent individuals. Then the offspring individuals are modified by a mutation operator. Usually, μ and λ are small integers. As with the GA, an individual represents a possible solution of an optimisation problem. But the ES uses the parameter representation in a forms of vectors of integer or real numbers. This vector is called "object-parameter". This vector together with another vector of real numbers (the "strategy-parameter") defines the data-structure for a single individual. The data-structure is usually referred to an ES-chromosome. The object-parameters contain the variables which have to be optimised, while the strategy parameters control the mutation of the object-parameters .

The "genetic programming" (GP) brings the idea of the GA one step further and evolves computer programs [Koz94]. Not only parameters of a problem but also the algorithm for the problem solving itself is subject of evolutionary changes. Most of the theory behind GP is the same as that behind the GA. The main difference between GP and the GA lies in the representation of a solution. The GA operates on a string of bits that represents the solution. While GP creates computer programs as the solution. The main disadvantage of GP is the huge computing resources required to solve any real world problem.

For the optimisation of complex systems the combination of several optimisation methods

can be a promising approach. If several different evolutionary algorithms are combined, it is called the "application of different strategies". The "competing subpopulations" involves simultaneous using of different optimisation strategies. Both methods open new dimensions to the application of evolutionary algorithms and make one step towards the development of powerful tools for the solution of complex problems.

7.3 Practical Implementation of Optimisation

This section describes the optimisation program which was developed by the author of this thesis to perform experiments presented in this work.

A schematic of the adaptive closed loop setup is depicted in Fig. 7.3. It consists of three main parts: a reflectron time of flight (Re-TOF) mass spectrometer, a pulse shaper, and an evolutionary algorithm. The pulse shaper is controlled by the evolutionary algorithm and creates pulses with a temporal structure depending on the applied phase mask. The shaped laser pulses are focused with a mirror onto a molecular beam inside an interaction region of the Re-TOF mass spectrometer which detects produced ions. The response of the molecular system is used as feedback signal for the evolutionary algorithm. The algorithm adapts the temporal structure of the shaped pulses in a loop to optimise a specified process. The optimisation is carried out until the desired experimental output is received.

A genetic algorithm implemented with LabVIEW is used in the present work. This algorithm was chosen because it outperforms other evolutionary algorithms in noisy environments [TLo94].

A population of individuals is realised as an array of structures consisting of two elements. A first element of the structure is a vector of integer numbers for the encoding of an individual (the phase mask used in the shaper), while a second element presented by a single real number characterises a fitness value of the corresponding individual (an expression derived from a combination of measured experimental signals).

The vector of 10 bits integer numbers (optimisation parameters) represents a spectral phase distribution (phase mask). Of course, the integer numbers have to be rescaled to the interval between 0 and 2π before be applied to the shaper. The maximal length of this vector is 640 and is limited by the numbers of pixels constituting the LCM. The vector length and

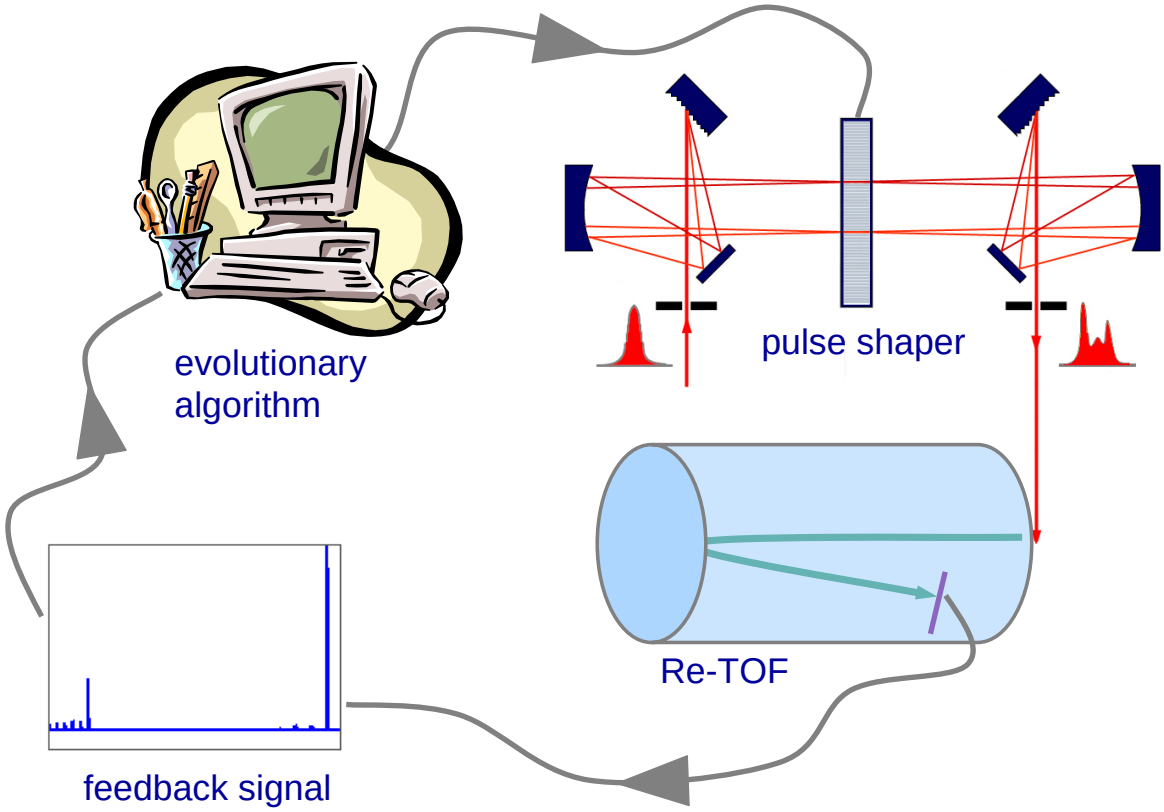

Figure 7.3: Scheme of the adaptive closed loop setup.

the maximal value of each parameter determine a size of the search space which in this case is 1024^{640}. This number is extremely huge. Therefore, to obtain reasonable convergence times the size of the search space must be limited. For that purpose the range of possible values of the optimisation parameters (often referred to as the resolution r) can be restricted to some number below 1024. Moreover, the vector length can also be reduced either by grouping of pixels or by parametrisation. In the first case a whole phase mask is reconstructed by using one parameter for a group of several pixels or by interpolation (linear or spline) of parameters to get a phase mask. In the second case a phase mask is described by some mathematical function and parameters of this function are optimisation parameters. The first case of optimisation is called "free optimisation", while the second case is corresponded to the parameterised optimisation. In the program used in this work, the genetic algorithm operates on 32 parameters only with 100 possible values of each parameter during a free optimisation. These two restrictions allow one to reduce a size of the search space down to 100^{32}. Taking into account that only the central fraction of the LCM is effective due to the narrow laser

bandwidth the real size of the search space is even smaller. But this is still a gigantic number. Thus, only random search methods, which are non-deterministic and stochastic, can effectively handle such large search space.

The number of individuals in the population has a great influence on the efficiency of the optimisation. The large number provides many degrees of freedom to explore the search space and minimise the risk of being trapped in a local maximum. This is especially important in the case of the complex search space with multimodal topology, such as encountered in the present work when e.g. optimising the fragmentation of model peptides. But the convergence time increases with growing population size as well. Generally, a larger population size is required if the optimisation problem has a high complexity. In this work, the population size of 20 individuals is determined from the condition that one optimisation run must not exceed two hours. This limitation mainly comes from the maximal time during which a stable laboratory environment can be maintained. Approximately the same size of the population is successfully used in many applications of pulse shaping even with a much larger search space [BER07, Mer07]. The two best individuals (survivors) are transferred into the next generation without any modifications (elitism). Generally, the population of individuals are initialised with random numbers. But the optimisation programm provides the possibility to include one or more preselected individuals in the initial population. Alternatively, the programm can try to build the initial population out the individuals which exhibit a fitness above some threshold. This strategy can be useful when, for example, an average fitness of a random phase mask is comparable with a noise level.

As described above, there are several methods to select individuals for the crossover. These selection schemes have various characteristics and therefore can influence the optimisation efficiency differently. For example, the proportional selection is not translation invariant [BHo91] and therefore the selection probabilities strongly depend on the scaling of the fitness function. The truncate selection excludes some number of individuals from selection. Moreover, the efficiency of a selection method can depend on the type of an optimisation problem. Here the ranking selection is employed as the main method because it behaves in a more robust manner [Why89]. After evaluation of the whole population of N individuals they are ranked according their fitness values starting from the best individual. A special parameter ν (nonlinearity) allows one to change the "nonlinearity" of this selection method and thus to regulate

the probability of better individuals being selected compared to the average probability of selection for all individuals. The probability of a m-th individual to be selected for the crossover is given by

$$p_m = \frac{1}{K}(N-m)^{[1-\lg\nu]};\ m \in [0, N-1] \quad , \tag{7.7}$$

where K is a normalisation coefficient which is determined from the condition of $\sum_{m=0}^{N-1} p_m = 1$. The parameter ν can be altered from 0 to 1. $\nu = 1$ corresponds to the situation of the linear ranking selection scheme. The reduction of ν shifts the selection probability to the direction of the individuals with the higher fitness. Finally, only one best individual is selected at $\nu = 0$. The crossover probability p_c itself is defined as

$$p_c = \frac{n-1}{n+1} \quad , \tag{7.8}$$

were n is a chromosome length. This manner of the crossover probability determination leads to very large values of the probability. For example, the crossover probability of $\approx 94\%$ corresponds to chromosome length of 32.

The non-binary representation of individuals is a key difference of this algorithm over the GA described in Sec. 7.2. Mainly, this difference effects on the implementation of the mutation. The mutation, which is applied with some probability p_m to each optimisation parameter of a chromosome, can modify a value of a parameter x according to the following expression

$$x \pm x_{max} P^{[1-\lg\sigma]} \quad , \tag{7.9}$$

where $P \in (0, 1)$ is a uniform distributed random number, x_{max} is a maximal possible value of x, and σ is a parameter which characterises the mutation step size. If the value of x obtained after the mutation is larger than x_{max} (or less than 0), it has to be rescaled to the range between 0 and x_{max} by the subtracting (or adding) x_{max}. The case of $\sigma = 0$ corresponds to the absence of mutation. The mutation variability increases with larger σ and it reaches the maximum at $\sigma = 1$ when any x is simply replaced with a random number.

The fitness function f is determined by the response of the molecular system upon excitation with shaped laser pulses. Depending on the aim of an experiment in the simplest case the fitness function can be equal to an integrated yield of some particular ion. Generally, the fitness is a function of several parameters. For example, if the yield of one particular ion a must be maximised, while at the same time the yield of different ion b must be suppressed, it

was found that an appropriate fitness function is

$$f = [a - a^0] \times [b^0 - b] \quad , \tag{7.10}$$

where a^0 and b^0 are the respective ion yields obtained with an unshaped laser pulse. Such definition of the fitness function allows one to maximise the formation of a particular ion and to minimise the formation of a different one simultaneously. After the evaluation of the current generation the best fitness f_{best} and the worst one f_{worst} in the generation can be found. The fitness averaging over the whole generation gives a mean fitness f_{mean}. An optimisation is run while f_{best} still increases. If no further growth of f_{best} is observed at least during 5 generations, the optimisation can be terminated. Usually, only a small number (between 20 and 30) of generations are required to find an optimal solution.

The choice of the parameters used in the optimisation program has a significant impact on its performance. Hence, it is very important to find a proper set of these parameters. The best way of doing so is to perform optimisations of real physical systems and investigate the influence of the parameters on the optimisation results. Since this requires a lot of time, a set of test optimisations was made instead. The main idea of the test optimisation is to imitate the pulse shaper by applying the Fourier transformation to a Gaussian spectrum (45 nm FWHM) modulated with a phase mask defined by the program. The goal of the test optimisation is to find a phase mask which minimises the deviation between the calculated shaped pulse and some predefined shaped pulse with a complex temporal structure. The pulse shown in Fig. 9.12d was chosen as a target. The fitness function f in this test optimisation is defined as

$$f = \int_{-\infty}^{\infty} |I_{target}(t) - I_{calc}(t)|\, dt \quad , \tag{7.11}$$

where $I_{target}(t)$ is the temporal structure of the target pulse and $I_{calc}(t)$ is the the temporal structure of the calculated pulse. The test optimisation was repeated 100 times for each set of the parameters to find the average values of the best fitness f_{best}, the mean fitness f_{mean}, and the worst fitness f_{worst}. The mutation probability p_m, the nonlinearity ν, and the mutation step size parameter σ were tested very carefully. First, the influence of p_m on the final fitness was investigated in a range of p_m between 0.001 and 0.9 with two kind of optimisations: a long and a short one. The long optimisation was performed with 1000 generations to see the maximal fitness achievable in such kind of experiment, while the short optimisation was done

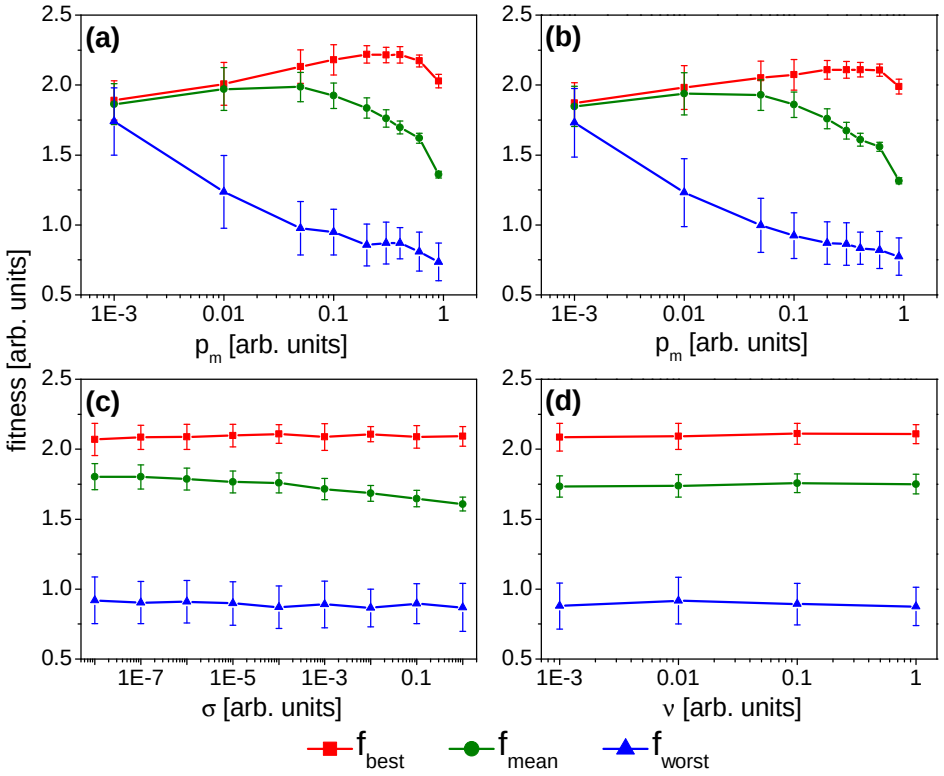

Figure 7.4: Results of the test to determine optimal values of the parameters used in the optimisation program: (a) test of the mutation probability p_m within 1000 generations; (b) test of the mutation probability p_m within 30 generations; (c) test of the mutation step size parameter σ within 30 generations; (d) test of the selection nonlinearity parameter ν within 30 generations. The best fitness f_{best}, the mean fitness f_{mean}, and the worst fitness f_{worst} are represented by red squares, green circles, and blue triangles, respectively. For details see text.

with 30 generations only to get the feeling about the fitness which can be received within a time limited real optimisation experiment. The results of these two optimisations are presented in Fig. 7.4a and Fig. 7.4b, respectively. The two important conclusions originates from these test optimisations. First, f_{best}, f_{mean}, and f_{worst} exhibit a strong dependence on p_m. f_{best} growths with increasing of p_m from 0.001 to ~ 0.3. At the same time the spread between f_{best}, f_{mean}, and f_{worst} is dramatically increased. f_{best} has a maximum around 0.3 ± 0.1 and drops sharply for larger values of p_m. Second, the difference between f_{best} obtained within 1000 and 30 generations is negligible small. Therefore, 30 generations are enough for the algorithm convergence. There is a recommendation to employ only one mutation

Table 7.1: Standard parameters for a free optimisation with the genetic algorithm.

Parameter	Value
Population size, N	20
Number of survivors, N_e	2
Number of optimisation parameters, n	32
Resolution, r	100
Selection nonlinearity, ν	1
Crossover probability, p_c	0.94
Mutation probability, p_m	0.2
Mutation step size parameter, σ	0.0001

per an individual [Bae93]. In case of 32 optimisation parameters the corresponding mutation probability is ≈ 0.03 only. Therefore, the mutation probability of 0.2, which is slightly less than the value from the test optimisations, is utilised in real optimisations. Then the mutation step size parameter σ and the selection nonlinearity parameter ν were tested within 30 generations. These results are shown in Fig. 7.4c and Fig. 7.4d, respectively. The influence of σ is very weak on the best fitness. The optimal value of σ can be seen around 0.0001. The effect of ν is not visible in the test optimisation. For the real optimisation this parameter is set to 1. Finally, the values of the parameters found with the test optimisations were verified experimentally with the optimisation of the second harmonic yield [Boy00]. Table 7.1 summarises all parameters used in the present work for a free optimisation with the algorithm described above.

In this chapter different optimisation methods were described. The practical implementation of the genetic algorithm used in the present work was given. Proper values of the most important parameters involved in the optimisation algorithm (namely, the mutation probability

p_m, the nonlinearity ν, and the mutation step size parameter σ) were determined using the test optimisation and summarised together with other parameters in Table 7.1.