Evolutionary Algorithms in Astronautic Applications

Draft: 8-Sep-10

Deutsches Zentrum für Luft und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Institut für Raumfahrtsysteme
Systemanalyse Raumsegment

Volker Maiwald

Robert-Hooke-Str. 7
D-28359 Bremen
Telefon 0421 24420-251
Telefax 0421 24420-150
E-Mail mailto: volker.maiwald@dlr.de
Internet: http://www.dlr.de/irs

Content

1 Introduction

Evolutionary algorithms (EA) are a computation tool that utilizes biological principles found in the evolution theory [1]. One major difference to other optimization methods is the fact that a large group of solutions is evaluated, not a single one. Combination of various solutions from such a group, called *population*, allows improvement of the solutions. Overall several terms in usage in the field of evolutionary algorithms have their origin in genetics or biology, especially the three major function principles of EAs: *Selection*, *recombination* and *mutation* [2].

Intrinsic to evolutionary algorithms is also the *fitness function*, which provides a numerical quality evaluation of a solution within the population of solutions [1] and thus sets the probability of this solution's reproduction [3]. Generally a fitness function is a function to be optimized by EAs. One major advantage of EAs in this respect is their ability to shift from one possible optimum to another and thus they are not bound to local optimization but can find global optima [1].

Regarding astronautic applications, evolutionary algorithms have been used for optimization of trajectories of low-thrust engines [4, 5, 6] and impulsive engines [7]. Various other fields apply evolutionary algorithms for optimization, e.g. aerodynamics [8] or warehouse planning [9].

This survey will concentrate on the usage of evolutionary algorithms for space applications, especially trajectory optimization and will try to describe future developments as currently planned and also to determine valuable areas of research.

2 Objectives and Proceedings

Evolutionary algorithms are applied in astronautics inter alia for trajectory optimization, but have been used in other areas as early as the 50s of the last century due to their ability to globally search a solution space. Their usage occurs in a very wide area of fields of research aside from space technology or even engineering [1]. Consequently advances in the subject of EAs also occur in various fields. This variety complicates the tracking of these advances and therefore the objectives of this work are the following:

- To provide an overview over EAs, their traits, their advantages over more other optimization methods and their disadvantages and drawbacks
- To examine the *global* trait of EA optimization and the methods used to improve this trait
- To grant an overview over the different directions that have been or are currently taken in further developing EAs
- To evaluate the applicability on trajectory optimization and determine valuable areas of further research with respect to this field

To achieve these objectives, first of all the concept of evolutionary algorithms is investigated. Furthermore the current application and methods used in combination with EAs are researched via a literature survey. It will be pointed out where improvements are possible and thus where further development and research is worthwhile.

3 About Evolutionary Algorithms

Evolutionary algorithms are a heuristic optimization method and generically describe computation methods that adapt principles from the evolution theory in optimization schemes. Subsets of these are *Genetic Algorithms*, *Genetic Programming*, *Evolutionary Programming*, etc. [10].
Genetic algorithms use data structures of fixed size and the development of the solution is done via mutation and recombination of previous solutions [1]. These algorithms will be concentrated on in this work.
Genetic programming uses data structures of variable size, mostly parse trees, [1] and/ or optimizes programmes [10].
Evolutionary programming leaves the view used in other evolutionary algorithms, i.e. regarding the solutions, which are evaluated, as individuals of *one* species, and broadens the view such that each solution is regarded as a *whole* species. Therefore several species are evaluated at once. Consequently no recombination between solutions is possible, but only mutation enables changing of the solutions [10].

As mentioned before, due to the nature of their optimization process, genetic algorithms will be concentrated on in this work. They are used for global optimization [11] and have certain traits in common [12]:

- Their data elements (solutions) are binary coded
- A whole population of solutions is subject to optimization
- Optimization is based on an objective function
- Transition is based on probabilistic rules and not deterministic ones

Especially the latter two allow computation of problems where little to no information is already present about the possible solution. What needs to be known is the desired result and the actual results are then evaluated according to their closeness to that.
Genetic algorithms are by their very nature valuable means to find a global optimum [12], yet there still is a certain risk that they converge to early to a local optimum. Various methods exist to prevent this; they will be elaborated in Chapter 4. The fact that several solutions are evaluated at once allows the combination of good solutions to find even better ones [2].
The remaining part of this chapter will lay out the theoretical background of genetic algorithms, including their basic building blocks and elements as well as processing functions.

3.1 Definitions

The following paragraphs will explain the terminology of the genetic algorithms, which heavily leans onto biology and anthropology. Not all terms are unique and in some cases several terms are used for the same idea – in that case it is attempted to use the most common term.

3.1.1 Chromosome and Genotype

The *chromosome* is the image of the data of each individual solution [1]; it is the solution's mapping [2]. A chromosome consists of one or more *genes* which are the binary strings of each design parameter. The individual positions within the string are called *loci* and their value is an *allele* [14]. Another term for chromosome is *genotype* [15].

3.1.2 Population and Generation

The *population* is the set of evaluated solutions [2], which are called *individuals*. The size of the population is an important issue to allow enough diversity of the algorithm to be successful but at the same time bears the danger of increasing computation effort. A population usually is initially filled with random solutions, which are then optimized. In case of large populations a lot of effort is needed to reduce the amount of bad solutions [1].
The population changes from iteration step to iteration step, to distinct between the different populations, the term *generation* is used [2].

3.1.3 Phenotypes

The *phenotype* of a chromosome is the expression, the appearance, result of its genes, in other words it is the chromosome's fitness value [15].

3.1.4 Optimization Objectives

Optimization is done with an objective – the measure that has to be optimized (e.g. flight time on a trajectory). This objective in turn provides the cost function or fitness function of an optimization. It is of course possible to use several objectives during optimization, different methods exist for formulation of a fitness function in that case. One, for example, weights the different objective function with factors and adds them to a cost function [18].
When optimizing more than one objective this usually results in a situation, where the improvement of one objective, decreases the solution's quality regarding another objective. Therefore a pareto-optimal solution is sought for. A solution is pareto-optimal, if it cannot be improved in one aspect without reducing its quality regarding another one [18].

3.1.5 Building Blocks and Schema-Theorem

The Schema-Theorem describes the probability of reproduction of schemata shared by several chromosomes. Schemata are defined by their *order* (o) and *defining length* (δ). The latter describes the distance between the first and last locus of a schema, whereas the former gives the number of identical alleles [16].

Example:

$$\delta(***1***0101001) = 10$$
$$o(***1***0101001) = 8$$

Equation (1) expresses the schema theorem mathematically and gives an estimate of the number of occurrences n of a schema S in the generation t+1. It depends on the number of occurrences in the previous generation t and the schema's fitness f related to the mean fitness $\overline{f}$. It furthermore depends on the probability p_c of crossover and its defining length δ as well as the probability p_m of mutation and the schema's order o [16].

$$n(S,t+1) \geq n(S,t) \cdot \frac{f(S)}{\overline{f}} \cdot (1 - p_c \frac{\delta(S)}{l-1} - o(S)p_m) \qquad (1)$$

Building blocks are schemata of short length and with above average contributions to good fitness. According to Goldberg [17] these building blocks combine to form good solutions of the optimization problem.
The reason behind this is that crossover is unlikely to disrupt schemata with a short defining length and mutation does probably not affect schemata of low order. Therefore the probability for their concurrent contribution to the solution improvement is higher than those of longer schemata or those of high order [16] (the extent depends on the probabilities for mutation and crossover).
The schema theorem is a simple formulation and does not cover all aspects of EAs, e.g. change of fitness and/ or mean fitness between the different generations. On the other side the most notable traits of EAs are very well covered – highly probable mutation reduces survival of high order schemas and highly probable crossover is likely to disturb very long schemas. Overall the settings of an algorithm should allow a balanced amount of exploration of the search space as well as sufficient exploitation of suitable and good solutions [16].

3.2 Basic Operations

Evolutionary algorithms are based on the evolution theory and the basic operations of *selection*, usually in some dependence on the solution's fitness, *recombination*, which can occur along different patterns, and *mutation*, which changes only parts of a chromosome.
Depending on the respective algorithm, see Chapter 3, and with weights on the single steps depending on the actual optimization pattern, the above mentioned algorithms share the following basic scheme for optimization:

1) Set-up of a population of possible solutions
2) Until a certain stopping condition is met, the following steps are repeated:

    i. Each individual solution is evaluated regarding its fitness to fulfil the optimization objective(s)
    ii. In dependence on this fitness individual solutions are selected for further processing with a certain probability
    iii. Recombine the selected solutions to new solutions with a certain probability
    iv. Mutate the solutions with a certain probability
    v. Insert the total of solutions into a new population

3.2.1 Selection

As seen in the example algorithm formulated before, the selection process determines the solutions, which are chosen for further processing, i.e. usually for procreation by recombination and if applicable further processing due to mutation. Selection is performed based on some scheme, which should allow a preferred survival of solutions with above average fitness [1], which is called selection pressure. At the same time this selection pressure should not be too large and thus hinder wide exploration of the whole solution space [16].

Furthermore there are selection schemes, which insert the new solutions instead of the original ones, i.e. with replacement, and those that do not replace the source solutions but other less fit solutions [10].

Selection Schemes
There exist various schemes for selection of the solutions, which are usually based on the general superiority of one solution, be it rank within the population or its fitness. The selection method has a significant influence on the overall algorithm performance [10].

One of the most common schemes is *roulette wheel* selection. In this case the probability of a given solution's selection is $p = f_i / F$, where f_i is the individuals fitness and F the total of all solutions' fitness. Obviously solutions on the lower end of the existing fitness values have a very low probability of being selected. However even the best solution is not selected with certainty [1], which results in less good performance when compared to *tournament* or *rank* selection [10].

In *tournament* selection the various solutions compete directly against each other based on their fitness values. This can occur in larger groups or in pairs. In the former case, k best solutions within a group are chosen for reproduction and their k children replace the *worst* solutions [1]. In the latter case, the mating pool is created by the winning solutions of the pairs. In all cases the method guarantees that the best solution moves over to the next generation [10], a process which is called *elitism*. Extensive prejudice favouring good fitness values, i.e. extensive *elitism*, does however hinder the wide exploration of the search space [1] and increases the risk of convergence to a local optimum [10].

Rank selection sorts all solutions into a ranked order, according to their fitness. The best solution has the highest rank, the worst solution has the lowest one. The probability for selection is then a function of the rank similar to roulette wheel selection [1].

Selection Noise
As described above the probability of selection for reproduction is increased for highly fit solutions, i.e. they are more likely to be explored further. However in case of equal fitness one solution usually receives biased processing, as the number of possible solutions is restricted. Consequently it can happen that very fit solutions drop out of evaluation before they have been thoroughly explored. This divergence from the predicted path of reproduction is considered noise [16].

Replacement
While no actual part of the selection process, the insertion of the new solutions into the new population usually follows comparable schemes, like roulette wheel replacement, rank replacement, absolute fitness replacement, random replacement. Either the children compete only against their parents or they are compared to the whole population [1].

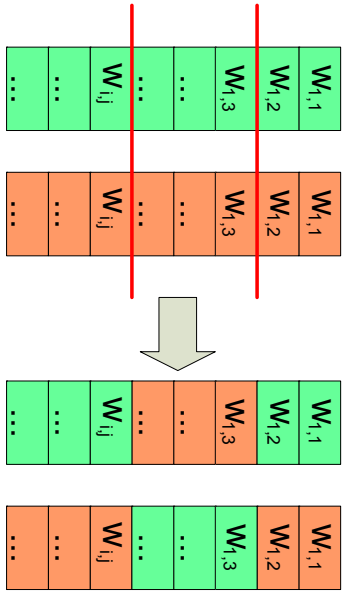
3.2.2 Recombination

The solutions, which are selected for further processing, are the parents of the new solutions to be evaluated. These new ones are usually referred to as children and are based on the original parents, they are recombinations of them. While it is possible to use totally different schemes, e.g. simply duplication (cloning) of existing solutions, or even random creation of new solutions, the reproduction of existing solutions is one core trait of EAs [10].

Recombination involves the cross-over of two (or theoretically more) parent solutions into a single new one. Especially if the children automatically replace their parents each pair of parents has to create two children. As for selection several schemes for crossover exist [1].

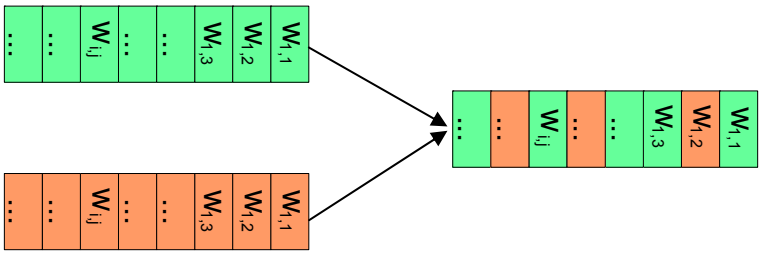
Point crossover, which is demonstrated in Figure 3-1, cuts the parents' chromosome at given loci and the children then share one parent's alleles before such a locus and the other one's after it. The two children are then complementary to each other. It is possible to use only one locus instead of several ones, in this case this is referred to as *single-point crossover*, in the other cases it is *multiple-point crossover*. Especially in the former one, it is difficult to save good genes to the child when these are distributed widely over the chromosome [1].

Figure 3-1: An example of a two-point crossover operation.



Uniform crossover, which is illustrated in Figure 3-2, randomly assigns one parent's allele to a child. For each locus it is randomly determined whether parent one's or parent two's allele is used [1].

Figure 3-2: An example of a uniform crossover operation.



Other methods exist, e.g. *adaptive crossover*. There each parent carries a template (consisting of 1s and 0s) along, which provides information on whose genes are used at what position. *Null crossover* means there is no crossover at all [1].

3.2.3 Mutation

Mutation is the EA process of introducing new data into the population by random change of single loci. This increases the diversity of the solutions and new areas of the solution space are investigated. The number of loci, which are changed, depends on the exact mutation scheme. Just as with recombination it is possible to change several loci, i.e. *single-point* or *multiple-point* mutation is possible. *Null mutation* means none is performed at all and *Lamarckian* mutation investigates up to *k* mutations and the fittest one is retained [1].

3.2.4 Assignment of Fitness

The fitness function is a numerical, heuristic evaluation of a given solution, assigning each individual within a population a fitness value. Higher fitness values should usually result in higher probabilities for selection and thus solution processing to exploit evolutionary effects. If a function is optimized it usually is set as the fitness function and a solution's function value is that case's fitness [1].

3.3 Advantages and Disadvantages

3.3.1 Advantages

Global search characteristics are probably the most beneficial advantage of EAs, caused by the parallel evaluation of many solutions within a population. They are furthermore able to shift from one optimum to another by introduction of new solutions into the population (either due to mutation or recombination) [1]. There are several techniques, which elaborate the diversity traits of EAs even further [16]. No additional information like derivatives is required for the optimization process, which makes them usable for trajectory optimizations where the final result is not known at the beginning of the optimization process [6]. Generally the application is not very complicated [19].

3.3.2 Disadvantages

One of the disadvantages of EAs is that it is difficult to find suitable stopping conditions for the optimization. The difficulty arises from the fact that the optimal result is usually not known and therefore it is problematic to determine when it is reached. Common stopping criteria are thus usually e.g. insufficient change of the solutions for a certain number of iterations [1].
In general, in EAs diversity always competes against convergence. Strong convergence to a solution results in only locally optimal solutions, the completeness of the solution space is only insufficiently searched. On the other hand, too strong diversity significantly decreases the search strategy's performance [16]. Overall the solutions are only compared among a randomly chosen set and not the complete solution space is investigated. Discretisation of the problem results in loss of gradient information and thus reduces the performance of EAs when compared to calculus-based methods [19].
Dependent on the exact problem, EA solutions are not very precise regarding the closeness to the global optimum [6].

4 Current Application and Developments

As mentioned before, currently one major utilization for evolutionary algorithms in astronautic applications is the optimization of (low-thrust) trajectories. To improve the performance of EAs especially regarding solution diversity there are several modifications of the standard algorithms used and in development. This chapter attempts to summarize current usage and also evolution that is still in progress.

4.1 Application

4.1.1 Combination with Artificial Neural Networks

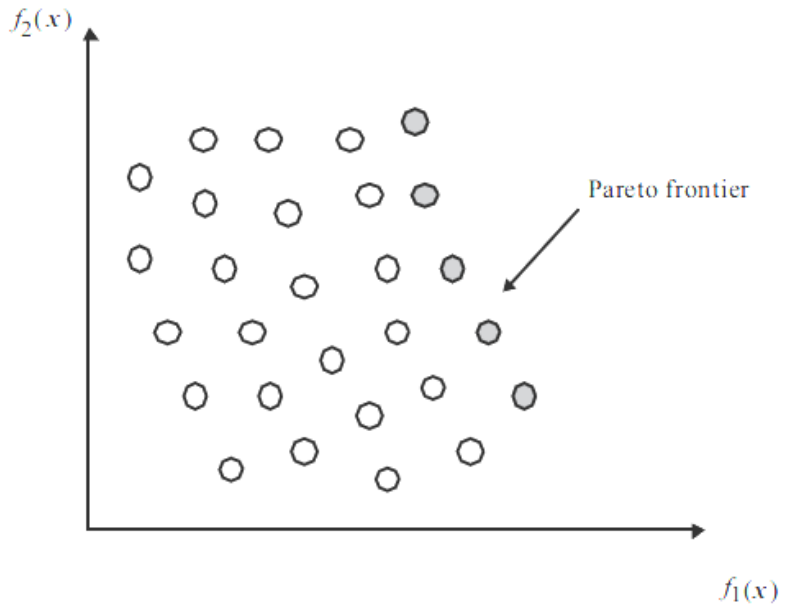
Application of evolutionary algorithms for use with artificial neural networks is not a new concept and has been performed for several applications, e.g. the simulation of virtual robots and their behaviour [1] or optimization of production processes [3].
Artificial neural networks are modelled after their organic paragons. They consist of computation elements, neurons, and weighted connections between these. The weights are the long-term knowledge of the neural network. Different structures exist. Usually such networks are organized in layers, input-, output- and hidden layers in case of feedforward networks, where information is passed on only in one direction. Usually only the weights are changed during optimization to adapt the network to the bestowed task [20]. There are different learning schemes, for trajectory optimization reinforced learning applies.
In difference to supervised learning, where an optimal solution is presented as example that has to be achieved, such an optimal solution is generally not known for trajectory optimization. Thus in a reinforced learning scheme, the solution's quality is evaluated on the criteria to be optimized. The network then adapts to improve the quality [20].
Dachwald [6, 21] and Carnelli, Dachwald and Vasile [22] have used a combination of neural networks and evolutionary algorithms for trajectory optimization with and without respect to gravity assist manoeuvres. The basic approach of the code *InTrance* is to use the neural network as controller ("pilot") and to train it in finding an optimal control strategy to assume an optimal trajectory with respect to certain objectives like minimum propellant consumption, minimum time of flight (TOF), etc. and constraints due to e.g. rendezvous conditions. The network is modelled via a network function N_π with π_i describing the various weighting factors. Several inputs like the state vector of the spacecraft and target, as well as propellant mass are fed into the network, which then produces the control vector consisting of throttle and thrust direction. Both the network's input and output are used to integrate the equations of motion over an interval which in turn grant the state vectors for the next point in time. This process continues until the stopping condition (e.g. maximum number of integration steps) is reached. The network is then evaluated according to the trajectory quality and receives a fitness value, while its network function is saved in a chromosome containing all weighting factors. This chromosome becomes part of the processing via evolutionary algorithms [6].
This method has been able to improve actual, existing trajectory designs regarding TOF, etc. [21] and is thus considered a successful approach on trajectory optimization.

4.1.2 Multi-Objective Optimization

Inherently trajectory optimization is a multi-objective optimization problem and these multiple objectives can contradict each other (e.g. minimum propellant mass and maximum payload mass). One traditional method of bringing the different objectives together for fitness evaluation is weighting of the objectives in regard to each other and combining the individual evaluations respective to each objective into one overall fitness value. However this usually requires extensive knowledge about the relations between all of the fitness contributors [15]. Another approach is to optimize only one objective and include the remaining ones as constraints for the optimization. In both cases, however, only a single solution is generated [21].

One method to create several solutions, which also show the dependence of the different objectives on each other, is Pareto-optimization [15, 21]. Pareto-optimality means that a solution cannot be improved regarding one of the objectives without decreasing its quality with respect to another one. In the objective space the so called non-dominated solutions form the Pareto frontier [9]. Non-dominated means that no solution has a better quality in all objectives, although there may be solutions, which have a better quality in some of the objectives [15]. The concept of a Pareto-frontier is illustrated in Figure 4-1 [9].

Figure 4-1: The pareto frontier of an optimization with two objectives $f_1(x)$ and $f_2(x)$ [9].



Srinivas and Deb have successfully designed a genetic algorithm that derives a solution's fitness directly from its rank in the Pareto frontier, where rank equals the row within the population [23]. It was also used by Hartman et al. for trajectory optimization [15].

This algorithm also includes modification due to sharing (see Chap. 4.2.1). Once the fitness according to rank is attributed to the first row, the evaluated solutions are removed from the pool and the remaining ones are then again evaluated until another Pareto-frontier is found. This receives a new fitness, reduced with respect to the first frontier's solutions. This process continues until each solution has received a rank evaluation and thus a fitness value [15].

4.1.3 Improvement of Convergence

As described in Chap. 3.3.2 one drawback of evolutionary algorithms is the low convergence rate when compared to local optimizers. While this allows a broad evaluation of the search space and in fact is mandatory for the global search as implemented for evolutionary algorithms, it still means that the final solution is not the actual global optimum but only close to it [24].

Therefore one aim of improvement of ordinary evolutionary algorithms is to improve their convergence via combination with e.g. local optimization strategies and algorithms. Such strategies are called memetic algorithms [24].

Crain, Bishop and Fowler [25] and also Wuerl, Crain and Braden [24] have successfully used a calculus-of-variations-method in unison with genetic algorithms. With that method the lack of solution precision of genetic algorithms and the lack of global search characteristic of local optimizers are remedied [24]. In a first step Crain et al. [25] used a genetic algorithm to identify the approximate region of the global optimum and once the found solution was within the globally optimal solution's probable radius of convergence a recursive quadratic programming method was used as local optimizer to find the precise global optimum. This way the genetic algorithm provided the initial parameter guess needed for the local optimizer, which usually has to be supplied by the mission analyst. The stopping criterion for the genetic algorithm was a preset number of generations (which should be at least equal to the chromosome size) and in this case was chosen as 40 by Crain et al. They furthermore report that a small population size could result in sensitivity of the algorithm regarding the initial random seed values. Generally the final results were as precise as with standard methods using local optimization, but due to the global search effort undertaken by the genetic algorithm, the number of trajectory evaluations has been two orders of magnitude smaller. According to Crain et al. [25] further improvement could be achieved by applying the local optimizer parallely to the genetic algorithm.

The results of this evolution have been reported by Wuerl et al. [24]. As before the final solution provided by the genetic algorithm served as initial input for the local optimizer. In addition two different learning strategies are optionally used during calculations (leading to increase in computation time). The first option is to check each solution for its potential in improving to a better quality – the fitness is adapted accordingly to represent "learned" traits that are not genetically inherited. The genome is not altered. In the second option the solutions are altered by a short local optimization (only for some iterations) and are also awarded a certain modified fitness. This is supposed to represent learned traits that are actually biologically inherited by children. This second option increased the convergence rate at the cost of diversity. Generally elitism, i.e. retention of the best solution, increases the performance of the memetic algorithm [24]. For both learning strategies a second stage of optimization has been added at the end of the genetic algorithms optimization, which was stopped once the convergence radius of the global optimum had been reached. The combined algorithm was able to reach this stopping criterion after around 20 generations for the applied problem (Mars to Earth trajectories). As before the overall results were at least as accurate as with other methods at an improved computation speed [24].

Vavrina et al. [13] have also incorporated a direct, gradient-based method (GALLOP) into a genetic algorithm search strategy. Their goal was to optimize gravity assist manoeuvres. However in their case every solution was locally optimized as an integral part of the genetic algorithm, after mutation and before fitness assignment.

Quite similarly Hartmann et al. [15] have successfully combined a modified genetic algorithm with a calculus-of-variations based optimizer called SEPTOP. Good performance of this optimizer requires initial guesses that are already close to an optimum. Thus the results of the global search via EA are used as such an initial guess to begin the local optimization. The

combination was able to produce more optimal solutions and also to produce solutions aside the initial guess of the operators [15].
Kim [26] states that simulated annealing is a global search technique with superior performance regarding genetic algorithms due to a larger convergence rate. Therefore in that work calculus-of-variations-based local optimizers are included in the process of optimization as several stages after the global search, which results in more precise solutions for minimization of time of flight calculations [26].

4.1.4 Multi-Level Optimization

El-Beltagy and Keane [27] have described several methods (not exclusively for EAs) for multi-level optimization, in which case level refers to the grade of accuracy. The strategies they reported include: sequential multi-level optimization, gradually mixed optimization and totally mixed optimization.
In sequential optimization, the level of accuracy is increased by staging several calculations of the same problem but increasing the accuracy with each calculation. The final population of a less accurate computation is used as starting population for the more precise calculation (cf. warmstart in InTrance). In case of gradually mixed optimization, more accurate and less accurate calculations are mixed during the whole optimization process according to a certain probability dependent on the number of evaluations already performed (later evaluations grow in probability to use more accurate parameters). The third method uses constant probability values for each level of accuracy throughout the calculations. In all cases genetic algorithms with niching functions (see Chap. 4.2.1) provided the best results [27].

4.2 Developments

4.2.1 Improvement of Diversity

In Chap. 3.3.2 it was already pointed out, that diversity plays an important role in EAs' capability to locate and determine the global optimum of an optimization problem. It furthermore is a contradiction to convergence. Fast convergence to an optimum means the algorithm does not completely search the solution space of said problem, i.e. it results in a small diversity. Generally the ability to maintain a global search pattern and to escape local optima relies on solution diversity [28]. In addition diversity and thus global search characteristics are the justification for the application of EAs. Maintaining diversity increases the probability to find a global optimum and is the only source of reliability of the algorithm regarding this goal, since the final result cannot be compared to a known global optimum (although of course an algorithm can be tested with problems where the globally optimal solution is known).
There are many different methods to improve diversity of genetic algorithms; the simplest one is already included in most EAs per design: mutation [1]. Another option is to run the algorithm sequentially or parallely for creating different, non-interacting populations or to introduce new random solutions into the population after a certain criterion is reached [16]. More methods have been developed.

Nested Partitions

Shi, Ólafsson and Chen [29] have proposed a new algorithm to be combined with – in their view – the local optimization scheme applied by genetic algorithms. Their new method is called *nested partitions*. In this case the (finite!) solution space is divided into one *most promising* region and the remaining region. This most promising region is subdivided into sub-regions. In each iteration step these subregions and the surrounding region are sampled for their quality by randomly selecting solutions from their domain and calculating their fitness values – the best of these becomes a region's *promising index*. If this sampling produces a new (sub)-region to be most promising it obtains this status for the next iteration. If the surrounding region is instead found to be more promising, the search returns to a previous level. In combination with the genetic algorithm, for each (sub-)region a population is sampled. Each population is optimized by the genetic algorithm normally with the constraint that each population may only draw solutions within its own region. Next each region receives an *overall fitness*, in this case determined to be the best fitness found within the region. Furthermore a new region is selected as being the most suitable and thus the space of possible solutions is reduced [29].

Niching and Sharing Functions

Ordinary genetic algorithms do not evaluate solution similarity [30]. Niching is used to represent the fact that certain species tend to specialize in certain regions/ environments [27], although one species may have several niches (e.g. representing feeding habits). The to be optimized function is considered as environment and niching is meant to ensure that the solutions are distributed evenly over this environment by penalizing the fitness value if there are several solutions within one region [3]. This even distribution is especially useful in case of multi-level optimization, where the evaluation's accuracy varies – generally more areas are investigated and the possibility to locate the global optimum improves [27]. Another reason for niching is the detection of multiple solutions (which can be close to the global optimum or to different optima) [16], which is especially true if the accuracy of the fitness function is not precise and therefore the solutions need to be robust [30]. Generally diversity has no end in itself, its goal has to be retaining good solutions and not just any solution, to analyse and explore all optima [16].

Even distribution is achieved by penalizing a given solution's fitness in case other solutions are close and thus do not add new information regarding diversity. Penalizing of a solution's (true) fitness $f_{i,true}$ in case there are solutions within a certain limiting distance, σ_{share}, is performed with the following function, giving the shared fitness $f_{i,\,shared}$ [27]:

$$f_{i,shared} = \frac{f_{i,true}}{\sum_{j=1}^{n} s(d_{ij})}, \qquad (2)$$

where n is the number of evaluated solutions in the population (note that j = i is also considered, thus the sum of the sharing functions is always at least 1 and thus the shared fitness always at best equal to the true fitness) and $s(d_{ij})$ the sharing function with [27]:

$$s(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right) & if \quad d_{ij} < \sigma_{share} \\ 0 & otherwise \end{cases} . \qquad (3)$$

The distance d_{ij} is a measure for the proximity of the solutions. For p-dimensional solutions and q optima, this distance becomes [27]:

$$d_{ij} = \sqrt{\sum_{k=1}^{p} \left(\frac{x_{k,i} - x_{k,j}}{x_{k,\max} - x_{k,\min}} \right)^2}, \qquad (4)$$

where $x_{k,i}$ resp. $x_{k,j}$ are the k-th parameters of the i-th resp. j-th individual and *max* resp. *min* denote the extrema of the allowable values.
The limiting distance is given by [27]:

$$\sigma_{share} = 0.5q^{\frac{-1}{p}}. \qquad (5)$$

It can be seen that for the definition of the limiting distance, knowledge about the number of optima (q) is required, which is usually not given for the mentioned problems – i.e. they need to be estimated.

The effect of lowering the fitness of similar solutions ensures a more even distribution of the solutions over the solution space. Solutions with good true fitness values that are penalized due to their close neighbours can still investigate their optimum further as their larger number increases reproduction probability. Generally the high fitness values tend to draw more population members to an optimum, the sharing function however acts as repulsive force against this attraction [3]. This way if a niche is too densely populated, wandering off into other regions becomes beneficial and therefore diversity is forwarded [16].

Crowding
Related to niching mechanisms are strategies involving crowding, which is also employed to increase solution diversity. The idea is that a solution replacement occurs based on similarity, i.e. distance, of the solutions to each other (see Equ. (4)) [16].
Crowding is achieved during solution replacement by selecting m possible candidates for replacement, which are in turn compared to the new solutions. The most similar solution is then (with a certain probability where applicable) replaced. If the possible candidates are selected from all solutions, the method is considered an explicit one [30].
Other methods, e.g. deterministic crowding, restrict candidates for replacement only to parents. As parents and children are per definition similar to each other, restricting replacement to each pairing (children of one pair cannot replace solutions not part of this pair) already ensures diversity [12]. This is called an implicit method [30].
Mengshoel and Goldberg have reviewed two different approaches to crowding, probabilistic and deterministic. The basic method of selection and replacement does not differ and is done in the before mentioned manner. What differs is the probability for replacement. In deterministic replacement the fitness determines whether or not replacement takes place. If the child is fitter than the parent, the latter is replaced by the former. Probabilistic crowding assigns each child and parent a probability, which depends on the fitness of the individual and the sum of the fitness of the competing child and parent [30]:

$$p_c = \frac{f_c}{f_c + f_p}. \qquad (6)$$

The difference in the replacement decision results in higher convergence rates for deterministic crowding along with the possibility to loose niches from the evaluation. Probabilistic crowding in turn keeps niches but has a lower convergence rate than the former, although the convergence is stable and predictable. The deterministic method is especially useful for flat fitness functions, due to the high convergence. Generally a mix of both methods allows exploitation of the benefits. In this scheme, which is called portfolio, each method is assigned a probability for use [30].

4.2.2 Gravity Assist Manoeuvres

Gravity assist manoeuvres allow increase of the ΔVs a spacecraft is capable of during a mission by exploiting close flybys of planets. Even the, regarding achievable ΔV, already well performing low-thrust missions can benefit from gravity assist manoeuvres and thus it is also of interest to include such swing-bys into the optimization of their trajectories.
The disadvantage that goes along with the increase in possible ΔV is however a decrease in mission flexibility as mission times need to be observed closely [26].
Crain et al. [25] have used a combination of a genetic algorithm and a local optimizer based on recursive quadratic programming for gravity assist manoeuvre optimization. The latter has been developed for optimization of gravity assist manoeuvres already, but only on a local scale. As sketched in Chapter 4.1.3 a sequential combination of the two optimization schemes was applied to optimize previously user-given sequences for gravity assist manoeuvres (Earth-Mars-Earth and Earth-Venus-Earth).
Debban et al [31] also optimized preselected gravity-assist sequences yet with a different method (not using genetic algorithms, but also a broad search involving a cost function). In their case low-thrusts were approximated as multiple high thrust manoeuvres and furthermore the trajectories were optimized using a shape-method for broad searches, which later on in the process are refined. The broad search applies only a two-body model that also uses a patched-arc approach. This approach employs Keplerian conic sections for coasting and for thrust arcs so called exponential sinusoids, which can be described by the following equation [31]:

$$r = k_0 e^{k_1 \sin(k_2 \cdot \theta + \phi)}, \qquad (7)$$

where k_0, k_1, k_2, θ and Φ are user-defined settings that affect the shape of and thus thrust on the respective trajectory leg. These arcs can be analytically approximated with Equ. (7). Evaluation is done via a cost function that regards propellant mass and the arrival v_∞ in approximation. Time of flight is not regarded and needs to be assessed by the user [31].
Good quality solutions are then subjected to optimization by GALLOP (Gravity-Assist Low-Thrust Local Optimization Program). During the optimization the gravity assist manoeuvre is modelled as instant rotation of the velocity vector and the trajectory is divided into legs, which range from one body to the next. These legs are further subdivided into equally long parts (length referring to time) and at the midpoint of the segment one ΔV is applied (the sum of ΔVs modelling the continuous thrust), between that position conic arcs are used as trajectory approximation. The leg furthermore has a so called match-point. The start of the leg is propagated forward to this point on the trajectory, the end is propagated backwards, until they match. A sequential quadratic algorithm is used for optimizing the spacecraft mass. The

method proves to be an improvement regarding calculation effort when compared with other methods of finding and calculating gravity assist manoeuvres. Overall the need for an astrodynamics specialist was reduced by Debban's et al. yet not eliminated as still a gravity assist sequence has to be initially chosen [31]. The same method was successfully used by Ham et al. [32] for investigating a nuclear powered low-thrust mission to Jupiter (JIMO).
Although not using genetic algorithms in their approach, Vasile and Campagnola also employed a combination of global search methods with local search to optimize gravity assist manoeuvres [33]. The global search was applied to evaluate a large number of trajectories for their quality and the best were then used as first estimate inputs for the local optimization. In their mission design, Vasile and Campagnola, optimized a transfer to Jupiter and following that a gravity-assist sequence of Jupiter's moons for a rendezvous with Europa. Different calculations were applied to obtain solutions for the first transfer and the subsequent gravity-assist sequence. The self-designed global search algorithms employed a simplified model, e.g. assuming coplanar body orbits and modelling of the low-thrust by using several impulsive manoeuvres and no actual continuous thrust. The direct method for local optimization applied a q-Gauss quadrature sum scheme. Their method was successful in determining a trajectory design for evaluation of mission feasibility [33].
Carnelli [33] and Carnelli et al. [22] have investigated the use of an evolutionary neurocontroller (ENC, s. Chapter 4.1.1) for optimization of gravity assist low-thrust trajectories. Carnelli assumed that artificial neural networks can exploit the possibilities of gravity assist manoeuvres for improvement of mission performance because earlier calculations have shown that they are able to exploit so called solar photonic assist manoeuvres in case of solar sail trajectories. These manoeuvres bring a spacecraft close to the sun to increase thrust before it moves further away [33].
Carnelli considered two different optimization schemes - first, the application of a single ENC for the whole mission or several ENCs, one for each trajectory leg. As the latter would have meant that the gravity assist sequence had to be submitted by the user of the optimization tool, he chose the first option. The calculations however were not able to improve or even reach the results of existing trajectories, which was partly due to the fact that rewarding gravity assist manoeuvres via fitness did not prove sufficient to benefit the reproduction of gravity assist trajectories. As only few trajectories are sufficiently close to the respective body's sphere of influence, their progress is not fast enough compared to trajectories not using gravity assist. Furthermore the effect of gravity assist manoeuvres on the trajectories is significant and very sensitive, which makes training of the neural network difficult and often even decreases the fitness of the trajectory, therefore causing such trajectories to be extinguished from the population. Furthermore, in difference to the solar photonic assist, where the source of fitness improvement does not change its position, in case of gravity assist manoeuvres, the source of additional "free" ΔV does change its position and therefore it is difficult for the ANN to learn how to exploit it efficiently. As an alternative to using an ENC for gravity assist optimization, Carnelli introduced a gradient based local optimizer into the process for single gravity assist manoeuvres, which proved successful due to the fact that the solution space is smooth and only has one optimum, when applying a so called b-plane. The b-plane is a plane perpendicular to the incoming trajectory, where it "pierces" the sphere of influence of the body. This gravity assist "consultant" for the network proved to operate successful in providing a suitable insertion into the body's sphere of influence [22, 33]. As alternative Carnelli suggests to use multiple ENCs, one for each trajectory leg, however this would require previous knowledge of the gravity assist-sequence [33].

5 Summary and Conclusion

In the previous chapters the basic evolutionary algorithms were presented along with their attributes and evolutions on this basic concept. The need for diversity has been explained and means to achieve it have been reported. Similarly other aspects such as improvement on convergence, multi-objective optimization, etc. have been part of this review.

It has been shown that combination of evolutionary algorithms with local optimizing methods (based on calculus-of-variations and gradient methods) does have the potential of significantly increasing the performance of an optimization tool. The solution quality of such combined methods is not below that of optimization with a non-combined tool, yet the number of necessary evaluations to find a global optimum have been reduced by two orders of magnitude for certain methods. It is therefore conclusive that a combined use of local and global optimization is highly beneficial and required to remain competitive regarding other tools. To successfully exploit the increase in performance it is necessary, to cease the global search as early as possible, i.e. to determine a point where local optimization suffices to find the global optimum (when its convergence radius is reached).
It is also clear that the improvement or maintenance of diversity is crucial for the performance of global search. Distribution of the solutions over the whole solution space ensures that all optima are equally explored and thus the global one can be found.
Due to the benefits of gravity-assist manoeuvres on mission feasibility there is also a great interest in exploiting these for low-thrust missions. Consequently also the interest in optimizing such trajectories is as great. The complexity of the problem has, up to now, however prevented the creation of simple methods for such mission designs. It is apparent that especially the determination of a suitable gravity assist sequence is difficult to achieve by optimization with genetic or global search algorithms alone. New methods and tools, e.g. the use of an expertsystem, are required to erase the need for an expert operator.

A Appendix

A.1 Trajectory Optimization

Trajectory optimization is the task of finding the *best* solution regarding a certain objective within a mission design. The solution contains a steering strategy that is *best* suited to achieve the mission under consideration of the given objective, which is usually evaluated at the end of the trajectory [35]. Depending on the methods used and the actual problem to be solved, it can be reduced to finding *any* solution at all for a given mission layout [21], however.

A trajectory maps a time interval $t \in [t_0, t_f] \subset \Re$ onto a state space $X \subset \Re^n$ [26], where n is usually 6 for spacecraft trajectories as the elements of X, $\{\underline{x}_{sc}\} \in X$, need to be able to completely determine an orbit, which requires six parameters. For trajectory calculations it is common to use the orbital position $\underline{r}_{sc} \in \Re^3$ and velocity $\dot{\underline{r}}_{sc} \in \Re^3$ [21]. Part of the trajectory calculation is also the control function $\underline{U}$ that maps from an interval, which is commonly identical to the one of the state space, onto a spacecraft control vector $\underline{u} \in \Re^m$. This control vector defines the thrust direction and throttle [26].

The trajectory is calculated by integration of the equations of motion, which can be expressed by six differential equations of first order of the following form:

$$\dot{\underline{x}}_{sc}(t) = \underline{F}[\underline{x}_{sc}(t), \underline{u}(t), t], F : \Re^{6+m} \rightarrow \Re^6 \qquad (8)$$

These differential equations are a dynamic constraint for the optimization problem [21]. A control that is able to find an optimal trajectory is referred to as optimal control vector $\underline{u}^*$. The task to find such an optimal control is referred to as optimal control problem [26].

A.1.1 Optimization Objectives

Each optimization is undertaken with regard to certain optimization objectives, a property that needs to be maximized (or minimized, which is exchangeable via the sign) at the end of the optimization process [35].

Usually the objective function, which mathematically describes the objective, has the form of [35]:

$$J = \Phi(\underline{x}_{sc}(t_f), t_f) + \int_{t_0}^{t_f} L(\underline{x}_{sc}, \underline{u}, t)dt, \qquad (9)$$

where Φ is a function of the final state and time, L is a general function of the instantaneous state, control and time, integrated over time along the trajectory.

For trajectory optimization there are two categories of objectives [21]:

1) minimization of transfer time
   i. for a given payload and propellant mass
   ii. for a given launch and propellant mass

2) minimization of propellant mass:
   i. for a given payload mass and transfer time
   ii. for a given launch mass and transfer time

It should be noted that launch mass is not independent of the payload mass and therefore there is no direct distinction between the two. On the other hand, the maximization of payload mass for a given launch mass (resp. minimization of launch mass for a given payload mass) is a typical system engineering optimization problem.
With regard to Equ. (9) the objective of minimization of flight time can be expressed as [26]:

$$J_T = \int_{t_0}^{t_f} dt = t_0 - t_f = T. \qquad (10)$$

The minimization of propellant mass yields [26]:

$$J_{m_p} = \int_{t_0}^{t_f} \dot{m}_p dt = m_p(t_0) - m_p(t_f), \qquad (11)$$

where $\dot{m}_p$ is the propellant mass flow and m_p the propellant mass.
The objectives of optimization usually contradict each other for trajectories – minimization of transfer time usually requires large thrust, which in turn requires large amounts of propellant. Therefore trajectory optimization is as multi-objective optimization. Methods of approaching this type of problems are described in Chap. 4.1.2.

A.1.2 Optimization Constraints

Besides the optimization objective the optimization is often subject to certain constraints that have to be fulfilled during optimization. There are various kinds of constraints; they can be distinguished in two types. One type of constraint, terminal, requires a condition to be fulfilled at a specific point in time, t_e, (e.g. proximity to a target body at the end of the trajectory) or it is evaluated during the whole trajectory [35].
The former can be expressed as [35]:

$$C = f_1(\underline{x}_{sc}(t_e), \underline{u}, t_e), \qquad (12)$$

where f_1 is the function that describes the constraint, while the latter can be provided by [35]:

$$C = f_1(\underline{x}_{sc}(t_f), \underline{u}, t_f) + \int_{t_0}^{t_f} f_2(\underline{x}_{sc}, \underline{u}, t)dt, \qquad (13)$$

where f_2 is another function that expresses the constraints mathematically.
Path constraints affect the whole trajectory or a section of it (e.g. that the consumed propellant mass may not exceed the loaded propellant mass) and can be formulated as follows [35]:

$$C(\underline{x}_{sc}, \underline{u}, t) \le f_1(\underline{x}_{sc}, \underline{u}, t), \qquad (14)$$

Each class of constraint can be an equality or inequality type of constraint. The former is common for terminal constraints, the latter for path contraints. Generally however they are

applicable for all categories. Equations (12) to (14) would have to be adapted accordingly [35].

A.1.3 Mission Requirements

There are several possible mission scenarios that can be subject to optimization. Their basic classifications are *orbit transfer problems*, *flyby problem* and the *rendezvous problem*. If required the scenarios can be extended to cover e.g. several flybys of different targets, etc. [26].
The mathematical formulation for each problem is [21]:

- *Rendezvous problem:* The control function $\underline{U}$ needs to provide a control vector $\underline{u} \in \Re^m$, which transforms the initial state vector of the space craft $\underline{x}_{sc}(t_0)$ to the state vector of the target $\underline{x}_T(t)$ while subject to the constraints of the equations of motion and the terminal constraint that $\underline{x}_{sc}(t_f)$ needs to be equal to $\underline{x}_T(t_f)$
- *Flyby problem:* The flyby problem is identical to the rendezvous problem, except that only the position components of the state vector need to be addressed.
- *Orbit transfer problem:* In this case a vector $\underline{Z}$ containing the orbit elements has to be transformed to assume the elements of the target orbit at the end of the trajectory.

A.1.4 Solving of Low-Thrust Problems

High-thrust trajectories that use few impulsive thrust manoeuvres usually have few dimensions regarding the solution space of an optimization. For a single manoeuvre, e.g. for a flyby, the manoeuvre at the beginning of the mission is defined by two thrust angles and the thrust magnitude. This means the solution space has three dimensions [21].
Low-thrust missions are characterized by long periods of continuous thrust. While the control variables in $\underline{u}$ remain the same the continuity of the thrust in time makes the solution space infinite [26].
Only in rare cases such problems can be solved [21], therefore a usual approach is to numerically discretise the problem in order to reduce the dimensions of the solution space. Therefore the continuous time interval $t \in [t_0, t_f] \subset \Re$ is divided into single points in time which then make up a discrete time interval $\bar{t} \in [\bar{t}_0, \bar{t}_f] \subset \Re$. Following this approach the optimal solution is no control vector $\underline{u}^*(t)$ but a control vector history $\underline{u}^*[\bar{t}]$, which has a finite number of dimensions, as the time interval is also finite [21].

A.1.5 Optimization Methods

As described in Chap. 4.1.3 the optimization methods for trajectories can be divided into direct and indirect methods, which are usually local optimization methods [21].
Direct methods directly provide solutions for the control variables, whereas indirect methods solve a Two-Boundary-Problem to gain the solutions indirectly [26]. The latter involves calculus of variations, i.e. calculation of the problem's Hamiltonian; the former is usually an approximation [35].
Techniques for direct optimization are Non-linear Programming and Collocation, which involves the discretisation of the timer interval as well [26]. It uses no gradient calculation but interpolation instead, which only gives approximate results [35]. Other methods are so called

shooting methods, where initial guesses are used to propagate trajectory solutions and evaluate at the end of the propagation. In this case the initial solutions need to be guessed by the optimizer [35], which makes profound knowledge of the problem and trajectory calculation in general a requirement for the user [21].

A.1.6 Equations of Motion and Propulsion Models

The equations of motion are based on Newtonian mechanics and can be determined, e.g. via the Lagrange equations to [26]:

$$\ddot{\underline{r}} + \frac{\mu}{r^2}\underline{e}_r = 0 \,, \qquad (15)$$

where $\underline{r}$ is the vector between the two masses of the system, $\underline{e}_r$ the unity-vector in its direction and μ is the gravitational parameter. In case of real problems, the right side has to be filled with terms for perturbations and thrust.
A thrust model for a solar electric engine can be expressed as follows [21]:

$$\ddot{\underline{r}}_{thrust,SEP} = \frac{1}{m}\underline{F}(\kappa, r) \,, \qquad (16)$$

where κ as the throttle parameter and $\underline{F}$ describing the thrust (in direction and magnitude) [21]. Combined, the complete model for a solar electric propulsion system would be:

$$\ddot{\underline{r}} = \frac{1}{m}\underline{F}(\kappa, r) - \frac{\mu}{r^2}\underline{e}_r \qquad (17)$$

Comparable equations can be derived for other thrust sources like nuclear electric propulsion or solar sails [21].

B Nomenclature

EA	Evolutionary Algorithm
ENC	Evolutionary Neurocontroller
EP	Evolutionary Programming
GA	Genetic Algorithm
	Gravity Assist
GALLOP	Gravity-Assist Low-Thrust Local Optimization Program
GP	Genetic Programming
LT	Low-Thrust
TOF	Time of Flight

C References

[1] D. Ashlock, Evolutionary Computation for Modeling and Optimization, 2006, Springer

[2] M. Pirlot, General local search methods, European Journal of Operational Research, 92, p.493-511, 1996

[3] S. Sette, L. Boullart, L. van Langenhove, Optimising a Production Process by a Neural Network/Genetic Algorithm Approach, *Engng Applic. Artif. Intell.* Vol. 9, No. 6, pp. 681~589, 1996

[4] Y. Li, H. Baoyin, S. Gong, Y. Gao, J. Li, 1st ACT global trajectory optimization competition: Tsinghua University results, Acta Astronautica 61 (2007) 735 – 741

[5] V. Coverstone-Carroll, J.W. Hartmann, W.J. Mason, Optimal multi-objective low-thrust spacecraft trajectories, Comput. Methods Appl. Mech. Engrg. 186 (2000) 387 - 402

[6] B. Dachwald, Optimization of very-low-thrust trajectories using evolutionary neurocontrol, Acta Astronautica 57 (2005) 175 – 185

[7] N. Assadian, S. H. Pourtakdoust, Multiobjective genetic optimization of Earth–Moon trajectories in the restricted four-body problem, Advances in Space Research 45 (2010) 398–409

[8] I. C. Kampolis, K. C. Giannakoglou, A multilevel approach to single- and multiobjective aerodynamic optimization, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2963–2975

[9] P.N. Poulos, G.G. Rigatos, S.G. Tzafestas, A.K. Koukos, A Pareto-optimal genetic algorithm for warehouse multi-objective optimization, Engineering Applications of Artificial Intelligence 14 (2001) 737–749

[10] T. Weise, Global Optimization Algorithms – Theory and Application – 2nd Edition, Ebook

[11] S. Settea, L. Boullart, An implementation of genetic algorithms for rule based machine learning, Engineering Applications of Artificial Intelligence 13 (2000) 381 – 390

[12] Pini Gurfil, N. Jeremy Kasdin, Characterization and design of out-of-ecliptic trajectories using deterministic crowding genetic algorithms, Comput. Methods Appl. Mech. Engrg. 191 (2002) 2141–2158

[13] M. A. Vavrina, K. C. Howell, Global Low-Thrust Trajectory Optimization through Hybridization of a Genetic Algorithm and a Direct Method, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, Hawaii, Aug. 18-21, 2008

[14] H. Mühlenbein, Parallel genetic algorithms in combinatorial optimization, Computer Science and Operations Research: New Developments in Their Interfaces, 8 - 10 January 1992

[15] J. W. Hartmann, V. Coverstone-Carroll, S. N. Williams, Optimal Interplanetary Spacecraft Trajectories via a Pareto Genetic Algorithm, The Journal of the Astronautical Sciences, Vol. 46, No. 3, July 1998, pp.267-282

[16] S. W. Mahfoud, Niching Methods for Genetic Algorithms, Doctoral Thesis, University of Illinois, 1995

[17] D. E. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addison-Wesley, 1989

[18] P.N. Poulos, G.G. Rigatos*, S.G. Tzafestas, A.K. Koukos, A Pareto-optimal genetic algorithm for warehouse multi-objective optimization, Engineering Applications of Artificial Intelligence 14 (2001) 737–749

[19] J.T. Betts, Survey of Numerical Methods for Trajectory Optimization, Journal of Guidance, Control, and Dynamics, Vol. 21, No. 2, March–April 1998

[20] A. Farooq, Biologically Inspired Modular Neural Networks, Dissertation, Virginia State University, 2000

[21] B. Dachwald, Low-Thrust Trajectory Optimization and Interplanetary Mission Analysis Using Evolutionary Neurocontrol, Dissertation, Universität der Bundeswehr München Fakultät für Luft- und Raumfahrttechnik Institut für Raumfahrttechnik, 2003

[22] I. Carnelli, B. Dachwald, M. Vasile, Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization, Journal of Guidance, Control, and Dynamics, Vol. 32, No. 2, March–April 2009

[23] N. Srinivas, K. Deb, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, Vol. 2, No. 3, p. 221-248

[24] A. Wuerl, T. Crain, E. Braden, Genetic Algorithm and Calculus of Variations-Based Trajectory Optimization Technique, Journal of Spacecraft and Rockets, Vol. 40, No. 6, November–December 2003

[25] T. Crain, R.H. Bishop, W. Fowler, Interplanetary Flyby Mission Optimization Using a Hybrid Global/Local Search Method, Journal of Spacecraft and Rockets, Vol. 37, No. 4, July–August 2000

[26] M. Kim, Continuous Low-Thrust Trajectory Optimization: Techniques and Applications, Dissertation, Virginia State University, 2005

[27] M.A. El-Beltagy, A.J. Keane, A comparison of various optimization algorithms on a multilevel problem, Engineering Applications of Artificial Intelligence 12 (1999) 639 - 654

[28] P. Gurfil, N.J. Kasdin, Characterization and design of out-of-ecliptic trajectories using deterministic crowding genetic algorithms, Comput. Methods Appl. Mech. Engrg. 191 (2002) 2141–2158

[29] L. Shi, S. Ólafsson, Q. Chen, A new hybrid optimization algorithm, Computers & Industrial Engineering 36 (1999) 409 - 426

[30] O.J. Mengshoel, D.E. Goldberg, The Crowding Approach to Niching in Genetic Algorithms, Evolutionary Computation Vol. 16 No. 3 (2008), p. 315 - 354

[31] T. J. Debban, T.T. McConaghy, J.M. Longuski, Design and Optimization of low-thrust gravity-assist trajectories to selected planets, Astrodynamics Specialist Conference and Exhibit, 5-8 August 2002, Monterey, California

[32] C.H. Yam, T.T. McConaghy, K.J. Chen, J.M. Longuski Design of Low-Thrust Gravity-Assist Trajectories to the Outer Planets, 55th International Astronautical Congress, IAC-04-A.6.02, Vancouver, BC, Canada, October 2004

[33] M. Vasile, S. Campagnola, Design of Low-Thrust Multi-Gravity Assist Trajectories to Europa, JBIS, Vol. 62, p. 15 – 31, 2009

[34] I. Carnelli, Optimization of Interplanetary Trajectories combining Low-Thrust and Gravity Assists with Evolutionary Neurocontrol, master thesis, Politecnico di Milano Facolt`a di Ingegneria Dipartimento di Ingegneria Aerospaziale, 2005

[35] S. Kemble, Interplanetary Mission Analysis and Design, Springer, 2006