Chapter

# An Overview of Evolutionary Algorithms toward Spacecraft Attitude Control

*Matthew A. Cooper and Brendon Smeresky*

## Abstract

Evolutionary algorithms can be used to solve interesting problems for aeronautical and astronautical applications, and it is a must to review the fundamentals of the most common evolutionary algorithms being used for those applications. Genetic algorithms, particle swarm optimization, firefly algorithm, ant colony optimization, artificial bee colony optimization, and the cuckoo search algorithm are presented and discussed with an emphasis on astronautical applications. In summary, the genetic algorithm and its variants can be used for a large parameter space but is more efficient in global optimization using a smaller chromosome size such that the number of parameters being optimized simultaneously is less than 1000. It is found that PID controller parameters, nonlinear parameter identification, and trajectory optimization are applications ripe for the genetic algorithm. Ant colony optimization and artificial bee colony optimization are optimization routines more suited for combinatorics, such as with trajectory optimization, path planning, scheduling, and spacecraft load bearing. Particle swarm optimization, firefly algorithm, and cuckoo search algorithms are best suited for large parameter spaces due to the decrease in computation need and function calls when compared to the genetic algorithm family of optimizers. Key areas of investigation for these social evolution algorithms are in spacecraft trajectory planning and in parameter identification.

**Keywords:** trajectory optimization, spacecraft control, artificial intelligence, genetic algorithm, particle swarm optimization, ant colony, artificial bee colony, cuckoo, firefly, swarm intelligence, evolutionary optimization

## 1. Introduction

Evolutionary algorithm use has been steadily increasing in the number of published papers corresponding to an increasing number of applications over the past 20 years [1–8]. Originating as an alternative to traditional mathematical optimization techniques, the techniques now span across almost every discipline to include data compression, traveling salesmen, image processing, and more importantly for spacecraft: control theory [9–13], system identification [14–19], and trajectory optimization [20–25]. Randomly searching a solution space to perform a global optimization routine can be computationally expensive and time consuming. Traditional methods such as stochastic parallel gradient decent, newton's method, and quadratic programming [26] are mathematical methods which rely mainly on the "steepness" of the gradients, or of the corresponding derivatives to follow the solution set to a zero-crossing gradient value and a potential optimum value. These methods are not easily implemented in discontinuous, or highly non-linear systems with a time-dependence such as in trajectory generation, and system identification of complex systems. One can perform a systematic, or random, search across an entire solution space, but the complex nature of some applications can limit the optimization to solution sets of only those within a small parameter space. Performing an exhausting search across an entire solution space can be considered the "brute force" method where the routine tries every single possible solution until the optimization criteria are met. For systems with a large parameter space with many variables, the computational cost might be too burdensome to arrive at a solution in a timely manner, and is certainly not relevant for a real-time application outside of a few cherry-picked examples. However, if a problem can be defined in an approachable way, evolutionary algorithms can provide a simpler and quicker way to find a viable solution. Due to the nature of these derivative free metaheuristic random search algorithms, the global optima may not be found but a suitable local optima that meets the optimization criteria can.

The most prominent evolutionary algorithm is the genetic algorithm officially introduced by John Holland in his 1975 book titled "Adaptation in Natural and Artificial Systems" [27] and its primary variants involving the concepts of chromosomes, elitism, parallel populations [28–30], and adaptation [31–33] which are derived from the concept of Darwinian evolution of animal species across many generations, also known as natural selection. Genetic Algorithms will be discussed more thoroughly in Section 2. The sister approach to natural selection based evolutionary algorithms are social-evolutionary algorithms also known as swarm intelligence which will be discussed in Section 3. Swarm intelligence is also a derivative free metaheuristic random search algorithm but with a slight modification on both selection criteria and on the definitions that spawn the "evolution". Swarm intelligence optimization algorithms for astronautical applications can be sub-categorized into particle optimization and combinatorics. Particle optimization includes particle swarm optimization [34], firefly algorithm [35], and cuckoo search algorithms [36] which focus on a large parameter space with a correspondingly large solution space that may be impossible to evaluate with traditionally exhaustive optimization routines. The artificial bee colony optimization [37], and ant colony optimization [38] algorithms are designed for a smaller investigation swarm but can successfully navigate a problem defined as an infinite set of combinations such as the commonly referred to problem of a traveling salesman visiting a large number of cities.

## 2. Genetic algorithm

Genetic algorithms have been a staple of heuristic artificial intelligence approaches since its inception in the 1960s and later more formally introduced by John Holland in 1975 [27]. In the 1990s this kind of random search global optimization routine became more mainstream through the use of greatly increased processing speeds brought on by the personal computers any more importantly GPUs. Just like other evolutionary algorithms, genetic algorithms rely on a very specific parameter space defining the population characteristics, parent selection, and success criteria.

Biesbroek presents a parametric study on the fundamental parameters within a genetic algorithm application in [39] via three cases toward spacecraft trajectory optimization. The fundamental parameters include population size, mutation probability, and cross-over probability. **Figure 1** illustrates the baseline genetic algorithm structure.

The first major step is in seeding a population. Seeding a population is done by presenting the algorithm an initial starting point to grow a population from. An alternative here is to randomly generate a population. A Gaussian distribution is one common approach. The parent population size is generally dependent on how many parameters are to be optimized via GA.

Holland described the genetic algorithm as being comprised of building blocks [27], which was later rederived by Goldberg [40] who related the population size with the quality of decisions and predicted that for an initial population of $n$, $n^3$ building blocks are potentially available in the algorithm. A rule of thumb is for $m$ number of parameters the expected population required for convergence scales with the square root of the problem. More specifically, Harik showed that using probabilities, a more optimal population size can be described by Eq. (1) [41].

$$n = -2^{k-1} ln(\alpha) \frac{\sigma_{bb}\sqrt{\pi m}}{d} \tag{1}$$

where $k$ is the order of building blocks, $\alpha$ is the probability of GA failure, $\sigma_{bb}$ is the average noise expected on the quality of building blocks, $m$ is the number of the building blocks within the parameter minus one, and $d$ is the difference between the best and the second best fitness values. In this definition, the noise is a description of how the building blocks create a member via the genetic evolution of the algorithm and how the resulting combinations may interfere in finding the optimal solution. In other words, the parent population can create noise hindering the convergence efficiency. With the expectation on the population size that it scales with $\sqrt{m}$, it can be seen how quickly the required population and therefore computational need increases with the number of parameters. A more recently derived rule of thumb is that for "small" parameter sets, the population size is effective if scaled with the number of parameters with 10 m, and for larger spaces the population size scales with $ln(m)$ [42], where the definition of large is different for each author. For a simple parameter set GA can be quite effective for optimization

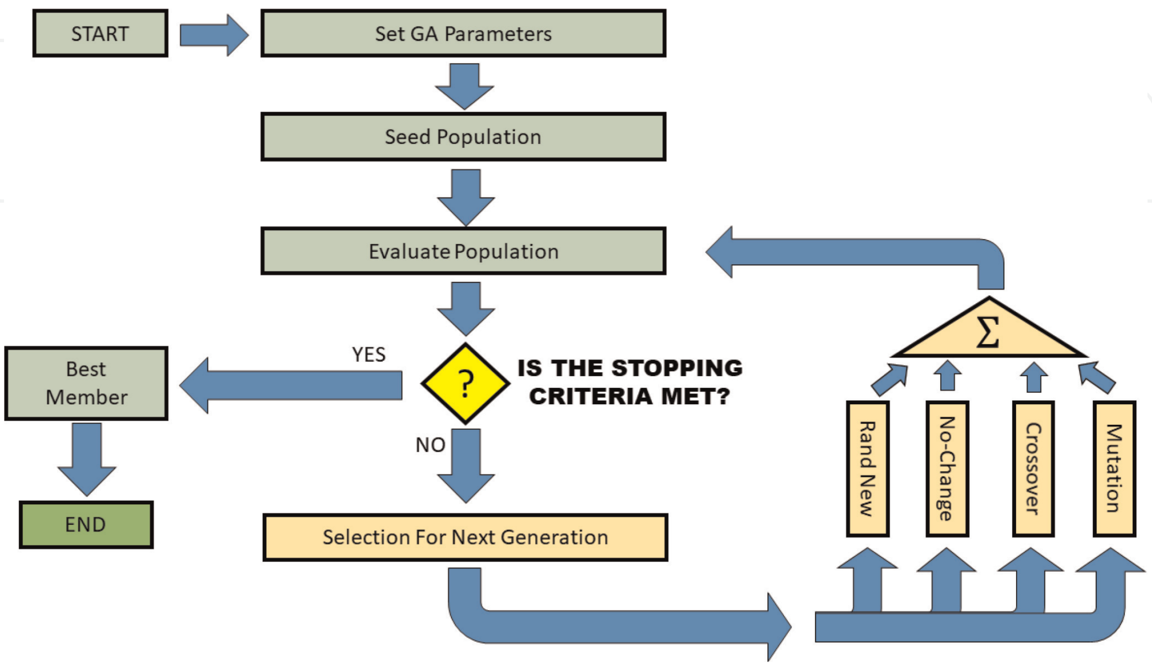

**Figure 1.**
*Generic genetic algorithm flow diagram.*

problems in many space based applications [23, 24, 28, 43–47] which stem from aeronautical control [20, 22], and ground-based robotic systems [48–50].

Looking again at **Figure 1**, the next step is to create children. Children are the result of statistical combinatorics of parents, and both of the mutation and cross-over of that parent population. The initial parent population will be evaluated and scored based on an objective function. The objective function is entirely user defined for the specific application. The objective function could be described with respect to minimizing the control effort needed to achieve a specified trajectory such as described in [51], or in minimizing the error between the actual and the desired trajectory in an attitude control scenario as presented in [52]. The objective function is the key component of any optimization and it is crucial to define it in such away as to minimize (or maximize) said function efficiently and precisely. This may seem obvious but the optimization routine will focus on the weighted parameter space defined by the objective function. If a control variable is not fully observable in the prescribed control law for a system, for example, the optimization may never converge to a viable solution set.

The objective function will be used to rate the performance of each member of the parent population. The population will then be ranked based on the threshold parameter specified in the algorithm. Different variations of GAs focus on certain threshold schemes to achieve faster convergence in various applications. In the basic GA scenario, those members who performed well enough to score below the threshold value (for a minimization problem) will form the parent population for the next generation. Additionally, a random subset of the original parent population will remain unmodified. Through mutation or cross-over based on mutation probability and cross-over probability. These parameters will define the statistical probability that a member of the population will be chosen for mutation or crossover. These probabilities typically start with a higher value and continually decrease on each subsequent generation to encourage the population to converge nicely to an optimal value. If a member is chosen for *mutation*, in this context, that will mean that a randomly chosen set of parameters within that member (if the parameter space is larger than one) will be adjusted via a Gaussian distribution function such that the amplitude of the specific parameter will have a statistical mean at the current value of the parameter. In short, a mutated member of the population will only have some parameters altered in value, not its entire chromosome, or set of parameters to optimize.

*Cross-over* is the next primary method by which the GA, alternatively described as a non-stochastic optimization approach, is taken. Non-stochastic optimization leads into the burgeoning field of deterministic artificial intelligence which entails more than there is space available in this chapter to discuss, and the reader is encouraged to review the following work compiled by Sands [53]. A cross-over is created by taking two parents, and through a predefined or randomly selected crossover point, they will be split and recombine as shown in **Figure 2** where Parent 1 and Parent 2 will now become Child 12 and Child 21 in the new population. After each member of the population is statistically chosen to be either modified or left alone it now is considered the children population. This new population is evaluated through the given objective function and the results become the segregated parent population for a new generation of possibilities. This cycle of parent-children function evaluations repeats until exit criteria are met. Common exit criteria may be to stop after a given number of generations have been evaluated, if the delta between the best performers across multiple generations shows no improvement, or ideally if the best performer of the current generation has met the performance objective desired. In this fashion, the population evolves over time via mutation and cross-over until an optimal solution is reached. In the best case, the optimal solution is

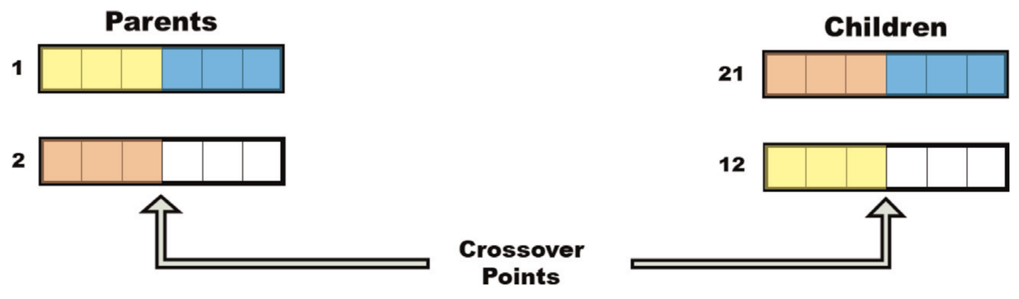

**Figure 2.**
*Generic genetic algorithm crossover illustration.*

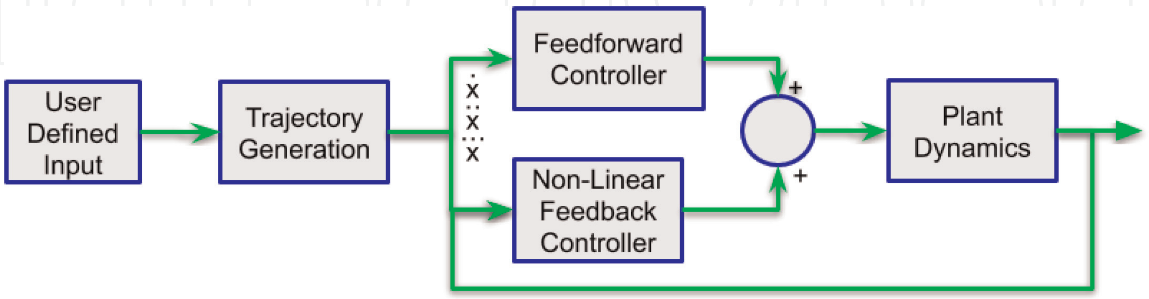

**Figure 3.**
*Sample control flow diagram using the non-linear Van der pol equation as a system under test.*

global, but is often local with the hyper-dimensional solution space too large to confirm one way or the other.

In the flow of the genetic algorithm framework, the algorithm is said to have "worked" when it reaches convergence. Convergence here is defined such that if the fitness of the entire population is decreasing (not counting stall generations where fitness is not improved) toward a global minimum, and that on the last generation the majority of the population has very similar fitness. An example is presented in **Figure 3** which illustrates a simple problem of tailoring the input of a trajectory system utilizing the dynamics based on the forced Van der Pol equation shown in Eq. (2). In this simple control example illustrated in blue and green, some arbitrary steady state value is the input to the system. It is then converted to a desired trajectory and sent to the feedforward controller. The feedforward controller builds a desired torque which is then sent as the control signal to the system dynamics, also know as the "plant". In an ideal situation, the equation used to determine the required control torque is perfectly understood such that the system is perfectly controllable. Here, the Van der Pol system converges to a desired state but only after through a large amount of transient states which can be detrimental to the physical stability of a mechanical system.

$$\frac{\partial^2 x}{\partial t^2} - \mu\left(1 - x^2\right)\frac{\partial x}{\partial t} + x = F(t) \tag{2}$$

$$K_i(x_m - x_d) + K_d(\ddot{x}_m - \ddot{x}_d) + \left(K_i(x_m - x_d)^2 - 1\right)K_p(\dot{x}_m - \dot{x}_d) \tag{3}$$

This control system breaks the desired input into three components, $x$, $\dot{x}$, and $\ddot{x}$ which represents angle, angular velocity, and angular acceleration, and are used to feed the feedforward controller of the system in order to prescribe the best torque command to the system dynamics described in Eq. (2). Using a non-linear PID feedback control scheme as seen in Eq. (3) where $K_i$, $K_d$, $K_p$ are the integral, derivative, and proportional gains, and $x_d$, and $x_m$ are the desired and measured output values respectively. Tuned by hand, one can achieve an RMS-error between the desired and measured circular trajectory in the phase-plane of this simulated system at $0.1767deg$. The $x$ and $\dot{x}$ components are shown in **Figure 4a**. This type of highly non-linear feedforward – PID feedback system is challenging to classically tune therefore a GA was investigated. In this scenario the objective function is set to minimize the RMS-error between the desired $x_d$ and $\dot{x}_d$ trajectory when compared to the actual trajectory $x_m$ and $\dot{x}_m$. The objective function can be seen in Eq. (4). Using a GA with the identified parameters in **Table 1** an RMS-error of $0.1060deg$ is achieved representing a 40% reduction in error.

$$RMSe = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\begin{bmatrix} x_d \\ \dot{x}_d) \\ \ddot{x}_d \end{bmatrix} - \begin{bmatrix} x_m \\ \dot{x}_m) \\ \ddot{x}_m \end{bmatrix}\right)^2} \tag{4}$$

The reduction in RMS error is illustrated in **Figure 4b**. The dotted line is the desired trajectory whereas the solid line is the actual trajectory achieved within the

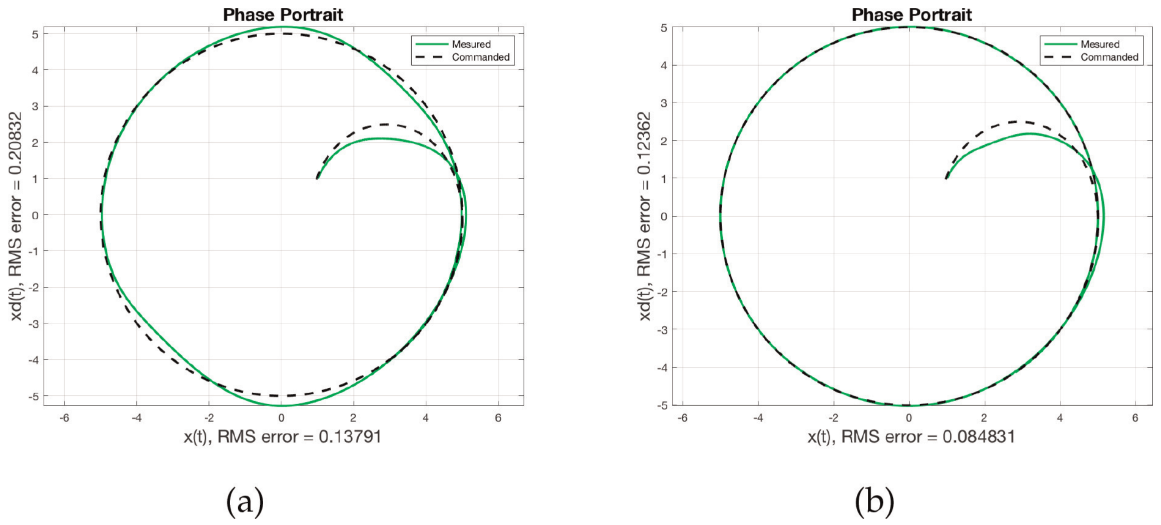

**Figure 4.**
*Trajectory results using classically tuned PID feedback control on a highly non-linear system versus the results using a genetic algorithm tuned to minimize the RMS error. A 40% reduction is RMS-error is achieved. (a) Trajectory results using classically tuned PID controller, and (b) Trajectory Results using a GA tuned PID controller.*

| Parameter | Value | Result |
|---|---|---|
| Population size | 200 | Function call per generation |
| Population | 3 | Number of parameters to optimize |
| Mutation rate | 10% | Probability to be selected for crossover |
| Crossover rate | 80% | Probability to be selected for crossover |
| Lower bound a | [0,0,0] | Minimum values allowed in population |
| Upper bound a | [1000,1000,1000] | Maximum values allowed in population |
| Selection criteria | 5% Elite | Choose the top 5% of population as parents |
| Max Stall generations | 20 | Exit criteria |
| Initial population range[a] | [0,20] | Initial population bounds |

[a]*Constraints to the genetic algorithm.*

**Table 1.**
*Genetic algorithm parameters for a highly nonlinear trajectory optimization of a Van der Pol system using PID feedback control.*

phase plane of the system. The phase plane plots the angular position vs. the angular velocity. The phase plane plot can be used to monitor trajectory tracking when you are interested in more that one state in addition to highlighting any potential instability in the system.

With this example, the rate of convergence can be illustrated and examined. **Figure 5** highlights the fact that the GA implementation presented here required 8800 function calls, and 43 generations to converge within the exit criteria at 20 stall generations. **Figure 5** also implies that reasonable performance was achieved after only 10 generations.

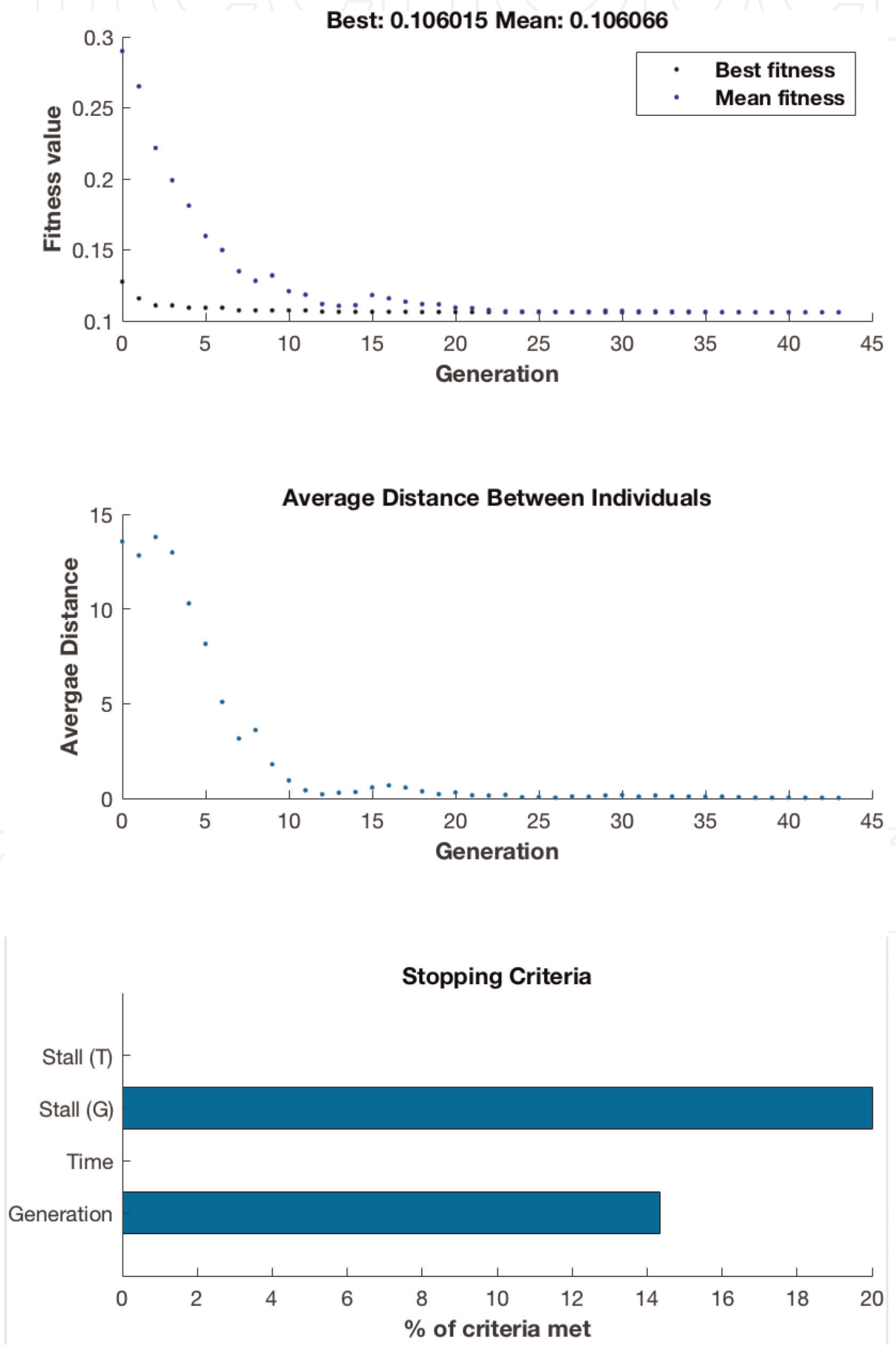

**Figure 5.**
*Results of the genetic algorithm as it steps through each generation in optimizing a PID control variables.*

Let us look again at the paper presented by Biesbroek [39], they look at a parameter space of only two in the application of optimizing the trajectory of a rocket such that a maximum horizontal distance, $x$, is achieved. The specific parameters are $V\left(\frac{m}{s}\right)$ & $m(kg)$, where V = velocity in meters per second, and m = mass in kilograms. The equation of motion for the dynamics of the rocket is defined in Eq. (5) along with the deferential equation to define the range $x$ with respect to $V$, and $m$.

$$\dot{x} = V, \quad \dot{V} = \frac{T - D}{m} = \frac{V_E\beta - kV^2}{m}, \quad dx = -\frac{m}{kV}dV - \frac{V_E}{kV}dm \tag{5}$$

where $T$ is thrust, $D$ is the drag, $V_E$ is the exhaust velocity, $\beta$ is the mass ratio, and $k$ is a constant. In this instance, the goal of the objective function is simply to find the values for $V$ and $m$ based on the equations of motion for the rocket and the differential equation defining the range. Using the equations defined in Eq. (5), it is then rearranged via Green's Theorem into the following objective function $f$ in Eq. (6). Depending on the author, an objective function may also be referred to a fitness function when describing how well the converged solution "fits" the desired result via an error equation. A common approach is to use the root-mean-squared error (RMSe) between the resulting solution and the desired solution.

$$f = \sum_{k=1}^{n}\left[\frac{ln(V_k) - ln(V_{max,k})}{k} - \frac{V_E}{k}\right]\delta m + 7686.722 \tag{6}$$

Here, $V_E$ is again the exhaust velocity, $V_{max,k}$ is the maximum velocity, and $\delta m$ is the change in mass as the propellant is used. In this case $\delta m$ and $V_{max,k}$ are the constraints that bound the solution space. Running through varying populations, cross-over and mutation populations lead to the quickest convergence with a population size of only 10,343 generations, a cross-over rate at 10% and mutation rate of only 1%. The optimal solution for two parameter problem presented can be calculated analytically to be equal to a distance of 9839 km. The most efficient GA solution, in terms of number of fitness function evaluations and therefore computational speed, required 3421 fitness function calculations and came to the correct solution such that the rocket should start with a full thrust until no more propellant is available, followed by a coasting arc to achieve maximum horizontal distance.

Another area of interest is in path planning. Jia presents a parallel evolutionary algorithm solution for real-time path planning for unmanned aerial vehicles (UAVs) in [29]. The Path planning problem for UAVs start with an initial point $x_0$, then the UAV needs to visit a series of waypoints, or stops, along its route, avoids potential dangers, or no-fly zones, before traveling back to $x_0$ or to another final landing zone as illustrated in **Figure 6**. In this scenario, a cost function might be described in such a fashion as to relate the fuel cost, a penalty of losing the UAV to a no-fly zone or a crash, and a reward for finishing the mission. The concept here takes a traditional genetic algorithm approach but modifies it through the use of competitive parallel generations. Each generation is evaluated through the objective function, but only the population with the best fitness value lives on whereas the other populations are re-initialized for the next generation. In the case of populations with similar peak fitness values, the population with the best overall fitness is chosen. The non-deterministic nature of path-planning presents an interesting challenge. The variability can stem from weather conditions, probabilities of danger to the UAV, and probabilities of failure modes of the UAV itself. The goal of using parallel competing populations is to mitigate the possibility of premature

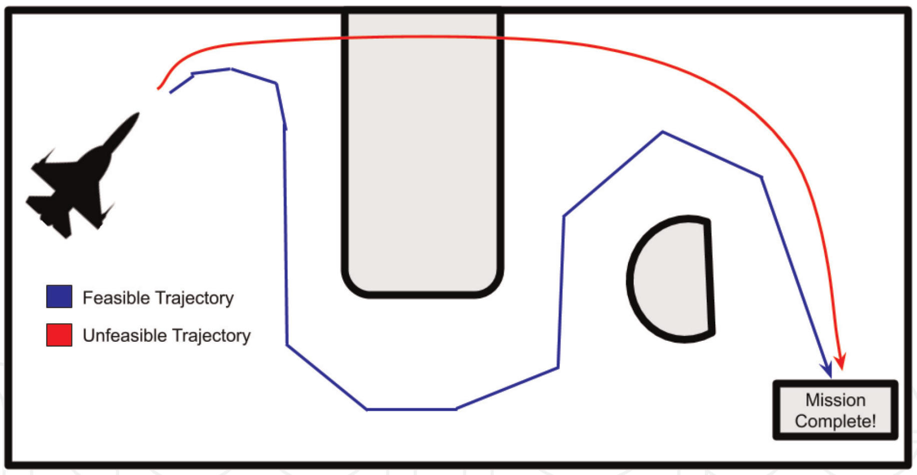

**Figure 6.**
*Simple illustration of a UAV path plan with no-fly zones in gray.*

convergence to a local minima. If the population with the best fitness stagnates in evolution to a local minima, the alternative parallel populations can potentially surpass the aforementioned population in fitness and therefore the stagnate population will become extinct and be re-initialized. This paper finalizes by highlighting the improvement of using two parallel evolutionary algorithm over a single in the ability to achieve a global minimum by 50% from 17%.

Taking the path planning application one step further leads us to the interplanetary path planning regime. Gage from Stanford University presented a novel paper in 1994 on interplanetary trajectory optimization using a genetic algorithm [43]. The preliminary design space for interplanetary spacecraft missions are highly complex, discontinuous, and anything beyond a two-body problem is primarily solved numerically. This initial investigation was on the viability of using a Genetic Algorithm as part of the team's suite of search methods toward finding viable interplanetary trajectories. It was shown that the computational need was reduced by almost four times from the more common grid-search method used up through the 1990s for designing interplanetary trajectories. The keys to the performance improvement were the constraints applied to objective functions which greatly penalized infeasible trajectories resultant from the parent-children, and to also artificially degrade the fitness of population members that were in close proximity within the GA's search space. This paper also noted that it can be helpful to restrict "mating" between parents with a separation $\leq \sigma$, where $\sigma$ is a user-defined threshold distance between potential parents within the same search space. This can ensure that parents that are circling around two different local minimums (not yet decided if one is a global minimum) do not mate and produce offspring that result near neither of the two local minimums, and thus are unlikely to increase fitness. This restriction in mating can lead to multiple possible solutions sets evolving through the generations and ensure that a single strong local minimum does not dominate the evolution.

## 3. Swarm intelligence

The next major subset of evolutionary algorithms relies on the assumption of swarm intelligence and social evolution in time whereas the genetic algorithms previously discussed depends upon genetic evolution of the populations; which is metaheuristic in nature and is derived from the biological (and probabilistic) mechanisms describing the movement of swarms of birds, fish, and insects on their search for food or mates. Referring to this concept as metaheuristic means that these algorithms are high-level symbolic based approaches designed to utilize imperfect or incomplete data in order to identify or approximate a sufficient solution to the given optimization problems. Swarm intelligence algorithms are based on the simple individual actions of the swarm which can collectively be quite complex and result in self-organization, decentralization and cooperation utilizing what is also referred to as collective intelligence. The primary swarm intelligence algorithms to be discussed are particle swarm optimization (PSO), cuckoo search algorithm (CSA), firefly search algorithm (FA), ant colony optimization search algorithm (ACO), and artificial bee colony algorithm (ABC).

### 3.1 Particle swarm optimization

Particle swarm optimization (PSO) was first introduced by Kennedy and Eberhart in 1995 [34] as a data clustering algorithm [1], and follows a population-based evolutionary social algorithm [3] along side of what can be considered an individualized random search algorithm [54]. **Figure 7** outlines the overarching procedure. Initially, the algorithm is seeded with a uniformly random distribution, which is called particles. These particles will result in many different values within the search space of the system of possibilities, and again, the individual particle performance is evaluated though an objective function just like with other optimization algorithms. In the general form of PSO, the algorithm utilizes a global topology which defines how the swarm communicates with each other. Utilizing the above mentioned communication, the swarm is aware of all other particle actions, success, and current velocity. Each particle's position within the search space (searching for the optimum value) is calculated with a corresponding velocity as defined by Eq. (7).

$$v_n(t) = \omega \times v_n(t-1) + a_1 r_1 \left(p_{B_n}(t-1) - x_n(t-1)\right) + a_2 r_2 \left(g_{B_n}(t-1) - x_n(t-1)\right) \tag{7}$$

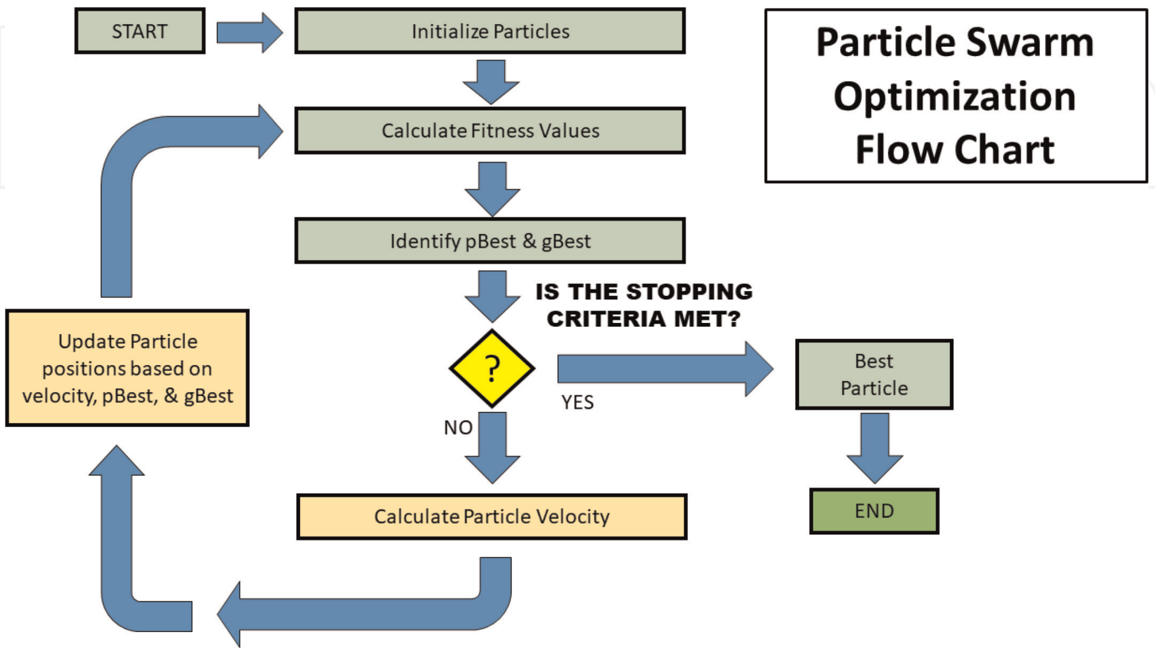
START
Initialize Particles
Calculate Fitness Values
Identify pBest & gBest
IS THE STOPPING CRITERIA MET?
?
YES
NO
Best Particle
END
Calculate Particle Velocity
Update Particle positions based on velocity, pBest, & gBest
Particle Swarm Optimization Flow Chart

**Figure 7.**
*Particle swarm optimization flow chart.*

where $v_n$ is the particle velocity in the next time step, $a_1$ and $a_2$ are constant weights on the effect due to social evolution, $p_{B_n}$ is the individual particle's best result so far, $g_{B_n}$ is the population's best result so far, $\omega$ is the particle's inertia, $r_1$ and $r_2$ are random numbers normally distributed from 0 to 1 to keep the random nature in the algorithm. Initially, the particles are generated with a small initial velocity and then grow as independent individuals within the swarm. PSO relies on the best particle solution position $p_{B_n}$, the best group solution position $g_{B_n}$, and the current particle velocity to calculate the particles next position in the solution space. Within that new particle position the objection function is then evaluated for fitness, along with all other particles in the population, and new $p_{B_n}$ and $g_{B_n}$ are calculated for the next iteration. The goal of particle swarm optimization algorithm is for all the particles to converge within the hyper-dimensional parameter space to the optimum value. With enough iterations, and a correspondingly well defined fitness function and PSO parameters, the algorithm will converge to global optima as illustrated in **Figure 8**. The algorithms parameters must be chosen such that the algorithm is balanced between two competing search notions: the particles exploring the unknown areas of the search space, and the particles exploiting the known areas of the search space in order to prevent premature convergence (if too heavily focused on exploiting) and non-convergence (if too heavily focused on exploring).

Alternatives and modifications to the PSO algorithm can include constrained velocities such as a minimum or maximum value (such that it will not grow unbounded), local biases defined by Euclidean distance between particles to define neighborhoods in order to prevent two sets of parents from different neighborhoods to attract to each other and reduce the overall fitness, and penalties for leaving the desired search space. Additionally, PSO can be hybridized with other approaches to utilize the lower computational cost of PSO but to decrease the randomness of the search if a general solution space is already known [55].

## 3.2 Firefly search algorithm

The firefly search algorithm (FA) is an interesting optimization technique based on the behavior of tropical fireflies who flash their abdomens with a bioluminescence chemical in order to both attract mates, and in some species to lure in prey such as the male of competing firefly species. With this behavior there are a few components that can lend itself toward the development of an interesting swarm optimization routine. Their light flashing pattern and intensity is also affected by their desire for mates or food, along with the distance another source is from the observing firefly. The definition on describing this pattern for what specifically

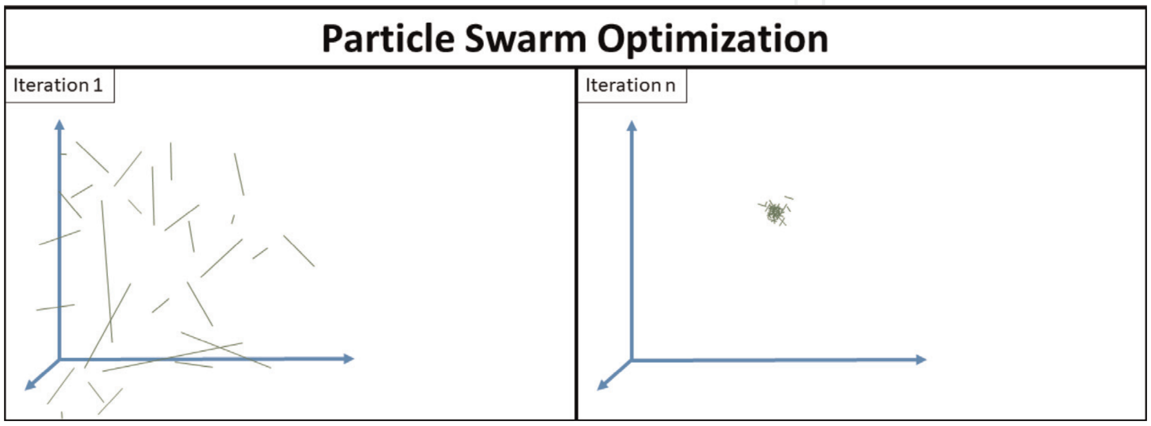

**Figure 8.**
*Particle swarm optimization iteration illustration.*

prompts the firefly to signal and how they signal there light is still unknown. But taking some of these concepts, an optimization routine can be developed. With this concept, the firefly algorithm was introduced by Yang [35]. The light intensity is a function of distance through the environment, which can be described by Eq. (8).

$$I_n = I_s e^{-\gamma r_{ns}^2}, \quad r_{ns} = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2}, \quad \beta \propto I_s e^{-\gamma r_{ns}^2} \tag{8}$$

where $I_n$ is the intensity seen by the nth firefly which is considered to be the observer, $I_s$ is the intensity of a nearby firefly also known as the source, and $r_{ns}$ is the Euclidean distance between the nth firefly and the nearby firefly. With this equation an attractiveness calculation represented by $\beta$ can be defined such that $I_n \propto \beta$ also shown in Eq. (8).

The FA routine is shown in **Figure 9**. This algorithm depends on an objective function just like the previously mentioned optimization algorithms. This objective function will evaluate the fitness of the fireflies, where the fitness will also determine the light intensity value at each firefly. An initial population of fireflies is generated and evaluated through the objective function. At this point the calculation for each firefly will be calculated to evaluate the attractiveness to its nearest neighbors. The attractiveness will affect the firefly's next step as shown in Eq. (9) [35].

$$x_{n_{t+1}} = x_{n_t} + \beta(x_n - x_s) + \varepsilon_n \tag{9}$$

where $x_{n_t}$ is the current position of firefly n and $x_{n_{t+1}}$ will be the new position of firefly n. $\beta$ is the attractiveness of firefly n to firefly s weighted by the distance $(x_n - x_s)$. $\varepsilon_n$ is a uniform distribution random number to facilitate a random walk behavior toward the more attractive mates. Starting from a relatively uniform population density across the solution search space, the fireflies will iteratively converge to the nearest minima location as seen in **Figure 10**.

The firefly search algorithm and its variants have been applied to trajectory optimization [56–58], control parameter optimization [59, 60], and dynamics

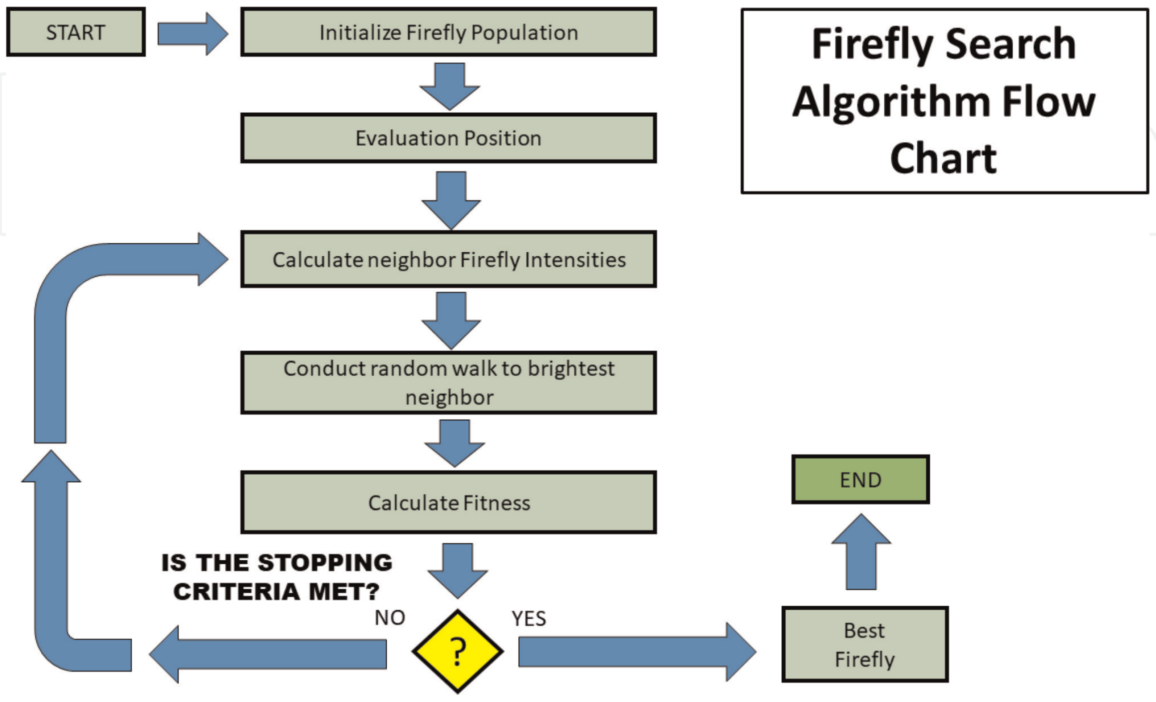
START
Initialize Firefly Population
Evaluation Position
Calculate neighbor Firefly Intensities
Conduct random walk to brightest neighbor
Calculate Fitness
IS THE STOPPING CRITERIA MET?
NO
?
YES
Best Firefly
END
Firefly Search Algorithm Flow Chart

**Figure 9.**
*Firefly search algorithm flow chart.*

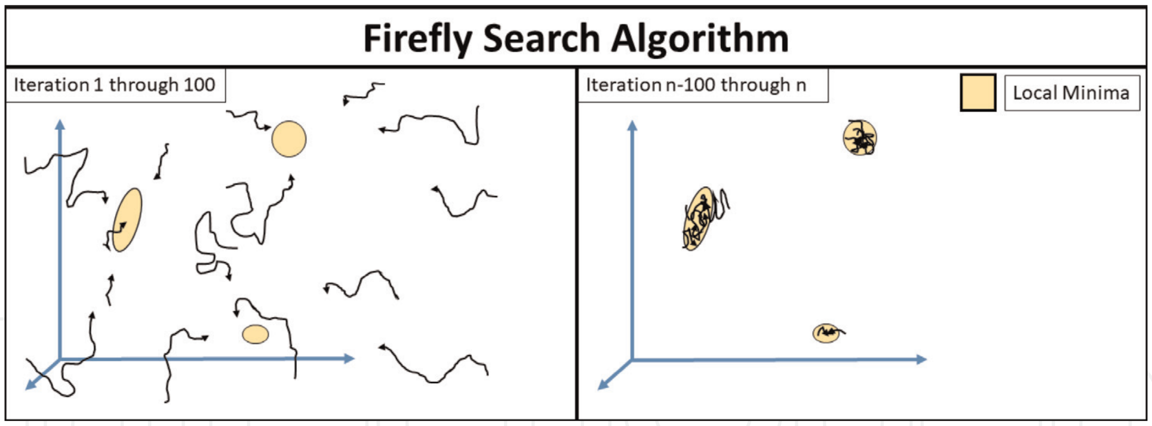

**Figure 10.**
*Firefly search algorithm iteration illustration.*

[61–63] in what can be considered as an introductory investigation by looking for initial successes in applying the FA toward these astronautical research areas.

### 3.3 Ant colony optimization search algorithm

The ant colony optimization (ACO) algorithm is another approach based off of the swarm behavior of insects, and was originally designed for combinatorial problems like the traveling salesman problem (TSP) where one tries to find the minimum path for an individual to traverse across n number of cities. In the instance of the traveling salesman, the salesman has a seemingly infinite number of combinations (but not really) to try but only wants to travel the shortest path, and to not repeat any stops along the way. The basic ant system, an earlier version of ACO, was presented by Dorigo in 1992 in his PhD thesis [38]. He presented a complex optimization algorithm based on the simultaneously simple and complex nature of foraging ants that would gain a large interest in the 1990s and later. Many variants and hybrids were presented by Dorigo and others. Most notably, offline pheromone updates, and pheromone evaporation was introduced which led to the more common ACO in 1999 [64, 65]. The concept of pheromones will be discussed shortly.

**Figure 11** illustrates at a high-level the flow of the ant colony optimization routine. The algorithm is initiated with a given set of parameters and objective function. Next, an initial set of solutions is populated. This is the first round of traveling ants looking for an optimal solution. The given problem is defined and broken apart such that the optimization routine will look for the optimal set of these building blocks in order to minimize the objective functions, or distance in the realm of the traveling salesman. At this point the pheromone level will be calculated. The pheromone is laid out such that each ant lays the same amount of pheromone out on the path that it traverses. This pheromone makes the links between different combinatorial building blocks attractive. If more than one ant traverses a similar segment the pheromone level will increase on that path segment, with an end result of the most common path being the most attractive.

With the most traversed paths being the most attractive, one may notice that there could be a strong potential for the algorithm to get "stuck" in a local minima. The ACO algorithm includes what is know as a local pheromone update, which means that either only the last segment of a successful path will include the pheromone, or that the end segment will be delivered a heavier weight of pheromone by the one ant who achieved the best path in the current iteration. Alternatively, the best path so far (out of all the iterations) may receive that additional pheromone instead. At this point the solution space is evaluated to see if a viable global solution

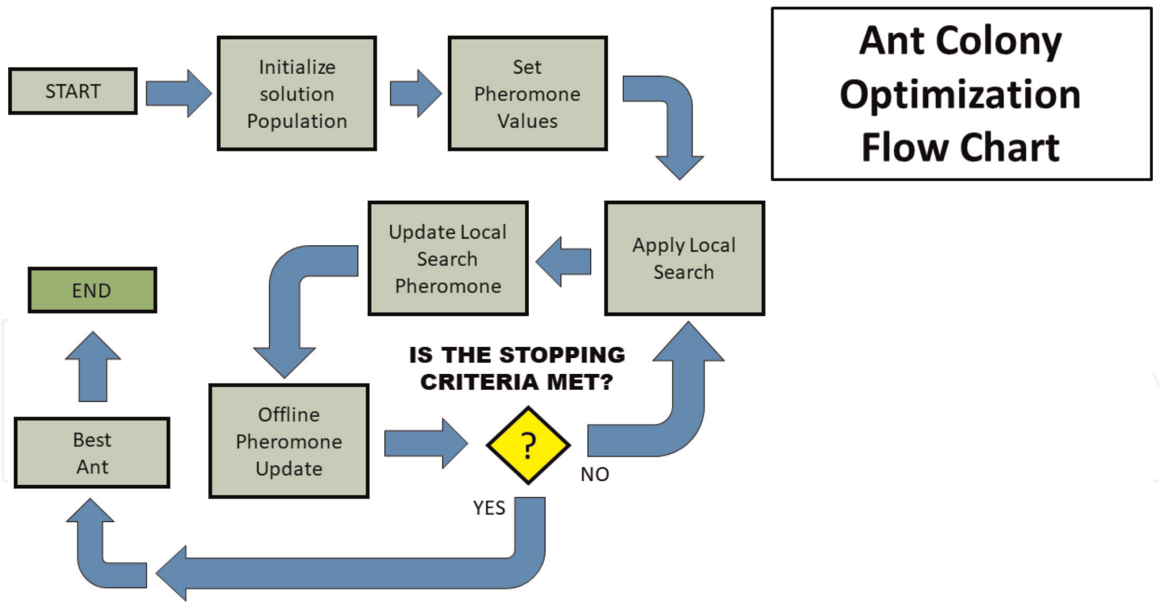
Ant Colony Optimization Flow Chart
START
Initialize solution Population
Set Pheromone Values
Apply Local Search
Update Local Search Pheromone
Offline Pheromone Update
IS THE STOPPING CRITERIA MET?
?
NO
YES
Best Ant
END

**Figure 11.**
*Ant colony optimization search algorithm flow chart.*

was found. If not, then the current ants will go through an exponential pheromone decrease before starting the next iteration in the optimization process. This pheromone decay reduces the impact of hysteresis on the solution space and helps prevent premature convergence.

On the following iteration the current ant population will then create a new set of paths to achieve a solution in the population. Each link will follow pseudorandom proportional rule, which uses a uniform distribution probability weighted by the segment pheromone values to decide on each link in the path. Once all ants create the new population of solutions another iteration of pheromone decisions will follow. This entire process will continue until convergence criteria are met.

Ant colony optimization has been successfully used for problem sets that can be discretized into a combinatory problem such as in path planning [66–68], trajectory optimization [21, 69], and even in spacecraft load bearing [70].

## 3.4 Artificial bee colony algorithm

Artificial bee colony (ABC) optimization is a derivative of both the bees system presented in 1997 by Sato and Hagiwara and the bee colony optimization by Teodorovic and Dell in 2005 [4], and was introduced later in 2005 by Karaboga [37]. The underlying algorithm flow chart is illustrated in **Figure 12** and involves three types of bees: onlookers, employed bees, and scouts. At the start of the algorithm, the routine parameters are initialized, and an initial population of food sources is generated via a uniform random distribution across the solution space. The population of food sources is discovered by employed bees and the quality of the food source is evaluated via an application specific fitness function. The employed bees then randomly search for a new food source, and if that food source is of better quality, then it becomes the primary food source. If not, then the new food source is abandoned. This is called a greedy search. Meanwhile, the onlooker bees observe the actions of the employed bees, and observe their communication dance through the lens of a randomly distributed variable. This represents the decision tree on which employed bee an onlooker will follow, or if it will create a new search. If the onlooker searches for a new food source, it will choose based on a random recombination of two solutions nearby. When a food source is not picked up by the onlookers when they transition to the employed bees, that food source is

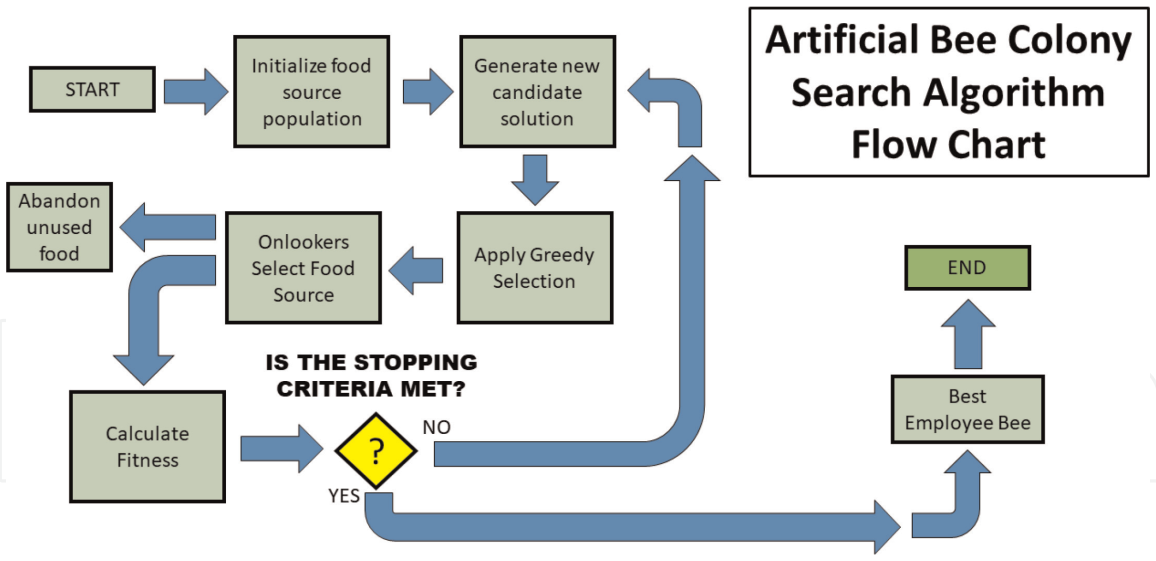
Artificial Bee Colony Search Algorithm Flow Chart
START
Initialize food source population
Generate new candidate solution
Apply Greedy Selection
Onlookers Select Food Source
Abandon unused food
Calculate Fitness
IS THE STOPPING CRITERIA MET?
?
NO
YES
Best Employee Bee
END

**Figure 12.**
*Artificial bee colony algorithm flow chart.*

now considered to be abandoned and will not be a part of the known current solution space. The next step is to repeat the search for new sources, a greedy selection, and the corresponding employed bee dance to randomly attract an onlooker to the known food source.

Similar to the ant colony optimization algorithm, the artificial bee colony algorithm is primarily designed for a combinatorics based problem where the potential solution can be discretized into an array of building blocks that can be rearranged and mutated to find an optimum solution. Additionally, the ABC algorithm has bee used for trajectory optimization [71], parameter optimization [72, 73], and remote sensing applications [74, 75].

### 3.5 Cuckoo-search algorithm

Another metaheuristic search algorithm is introduced with the cuckoo search algorithm (CSA). This approach mimics the parasitic brood behavior in certain species of the cuckoo bird. This type of bird has a fascinatingly aggressive reproduction strategy which is the heart of the CSA. Quite a few species of cuckoos participate in the obligate brood parasitism which means that they will lay their eggs in the nests of other birds [36]. The key to the success of the individual cuckoo is dependent on the cuckoo's ability to produce an egg that is able to mimic, or approximate, the host nest eggs such that the host nest mother bird can not distinguish the cuckoo egg from her own. The cuckoo egg will hatch before the host eggs, and ensure dominance in the nest and thus prolong their survival.

Taking this concept to an optimization algorithm it can be illustrated as seen in **Figure 13**. Step one is to initialize the algorithm and generate a population of host nests. Each host nest has a single potential solution to be compared against. The next step is to evaluate the initial population of nests with the defined fitness function. Once the fitness function is evaluated for each cuckoo egg, the next step is for the cuckoo to take flight via a classical Lvy flight path [36]. This is a type of step pattern is a heavy-tailed random walk similar to that of a fruit fly, which are observed to jolt out in a straight direction then randomly turn a sharp turn at a random angle. The desired behavior is described in Eq. (10).

$$x_{t+1} = x_t + sE_t \tag{10}$$

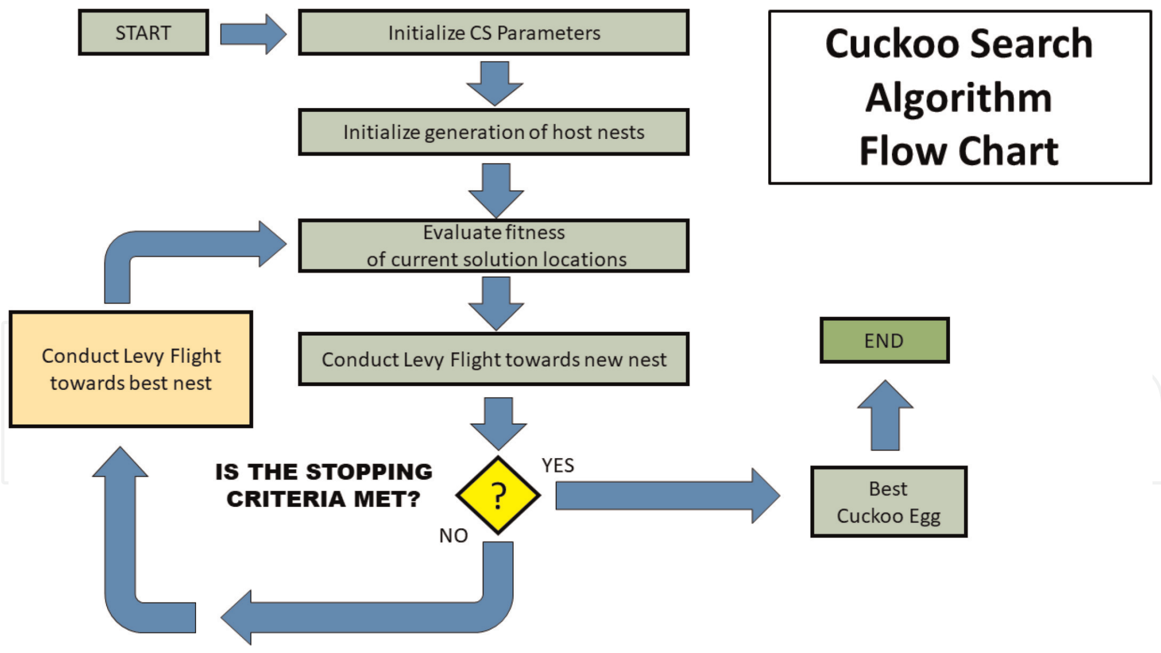
START
Initialize CS Parameters
Initialize generation of host nests
Evaluate fitness of current solution locations
Conduct Levy Flight towards new nest
Conduct Levy Flight towards best nest
IS THE STOPPING CRITERIA MET?
?
YES
NO
Best Cuckoo Egg
END
Cuckoo Search Algorithm Flow Chart

**Figure 13.**
*Cuckoo-search algorithm flow chart.*

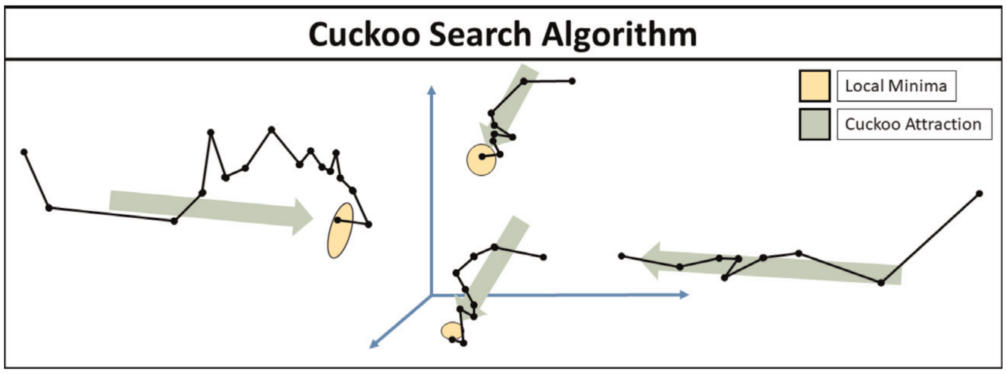

**Figure 14.**
*Cuckoo-search algorithm iteration illustration.*

where $x_{t+1}$ is the position in the next iteration, $s$ is the step size, and $E_t$ is a zero-mean random Gaussian draw to mimic Lvy flights. Once each cuckoo take a flight to a "new" nest, the fitness function is then calculated again to evaluate the new population of potential solutions. If the solution criteria are not met, then this process of flying from nest to nest will continue until convergence as illustrated in **Figure 14**. Each individual will take a random walk biases by the direction of the current best solution within the group.

Yang's and Deb's seminal work on the CSA illustrated an enormous computational cost savings when compared to the genetic algorithm and the particle swarm optimization Algorithm as each algorithm was used to find the solution to a handful of standard challenging mathematical functions: Michalewiczs, Rosenbrocks, Schwefels, Rastrigins, with a 96, 89, 96, and 91% decrease in the number of required fitness function evaluations when compared to a GA solution respectively [36].

## 4. Conclusions

With a prevalence of evolutionary algorithms focused on solving trajectory generation, path planning, remote sensing, control theory, and parameter identification for aeronautical and astronautical applications it is a must to review the fundamentals of the most common evolutionary algorithms being used for those applications. Genetic algorithm, particle swarm optimization, firefly algorithm, ant colony optimization, artificial bee colony optimization, and the cuckoo search algorithm are presented and discussed with an emphasis on astronautical applications. In summary, the genetic algorithm and its variants can be used for a large parameter space but is more efficient in investigating a smaller parameter space, less than 1000 parameters in the chromosome. It is found that PID controller parameters, nonlinear parameter identification, and trajectory optimization are applications ripe for the genetic algorithm. Ant colony optimization, and artificial bee colony optimization are optimization routines more suited for combinatorics, such as with trajectory optimization, path planning, scheduling, and spacecraft load bearing. Particle swarm optimization, firefly algorithm, and cuckoo search algorithms are best suited for large parameter spaces due to the decrease in computation need and function calls when compared to the genetic algorithm family of optimizers. Key areas of investigation for these social evolution algorithms are in spacecraft trajectory planning, and in parameter identification.

Evolutionary algorithms have been shown to have a great potential to solve challenging problems that traditional optimization routines may not be able to tackle due to large computational need to support an exhaustive search of the solution space. The reader should now have the tools to take the foundational material presented here, to review the referenced sources, and conduct their own deep dive in an application area of interest.

## Disclaimer

The views expressed in this paper are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or U.S. Government.

## Author details

Matthew A. Cooper[1]* and Brendon Smeresky[2]

1 Air Force Research Laboratory, Albuquerque, New Mexico, USA

2 Naval Postgraduate School, Monterey, California, USA

*Address all correspondence to: matthew.cooper.17@us.af.mil

## References

[1] Alam S, Dobbie G, Koh YS, Riddle P, Rehman SU. Research on particle swarm optimization based clustering: A systematic review of literature and techniques. Swarm and Evolutionary Computation. 2014;**17**:113

[2] Drugan MM. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. Swarm and Evolutionary Computation. 2019; **44**:228-246

[3] Ekici B, Cubukcuoglu C, Turrin M, Sariyildiz IS. Performative computational architecture using swarm and evolutionary optimisation: A review. Building and Environment. 2019;**147**:356-371

[4] Rajasekhar A, Lynn N, Das S, Suganthan P. Computing with the collective intelligence of honey bees a survey. Swarm and Evolutionary Computation. 2017;**32**:2548

[5] Shehab M, Khader AT, Al-Betar MA. A survey on applications and variants of the cuckoo search algorithm. Applied Soft Computing. 2017;**61**: 1041-1059

[6] Betts JT. Survey of numerical methods for trajectory optimization. Journal of Guidance, Control, and Dynamics. Mar 1998;**21**(2):193-207

[7] Gotmare A, Bhattacharjee SS, Patidar R, George NV. Swarm and evolutionary computing algorithms for system identification and filter design: A comprehensive review. Swarm and Evolutionary Computation. 2017;**32**: 6884

[8] Ball JE, Anderson DT, Chan CS. Comprehensive survey of deep learning in remote sensing: Theories, tools, and challenges for the community. Journal of Applied Remote Sensing. 2017; **11**(04):1

[9] Ahmed R, Chaal H, Gu D-W. Spacecraft controller tuning using particle swarm optimization. Proceedings of 2009 ICCAS-SICE. 2009. DOI: 10.1016/j.apm.2017.07.024

[10] Laware A, Talange D, Bandal V. Evolutionary optimization of sliding mode controller for level control system. ISA Transactions. 2018;**83**:199213

[11] Ghiglino P, Forshaw JL, Lappas VJ. Online PID self-tuning using an evolutionary swarm algorithm with experimental quadrotor flight results. AIAA Guidance, Navigation, and Control (GNC) Conference. 2013

[12] Gazizov RR, Zabolotsky AM, Gazizov RR. Use of genetic algorithm and evolution strategy when revealing the worst case effects of crosstalk propagation in PCB bus of spacecraft autonomous navigation system. In: Proceedings of 19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM). 2018. DOI: 10.1109/ EDM.2018.8435093

[13] Ge X-S, Chen L-Q, Liu YZ. Attitude control of underactuated spacecraft through flywheels motion using genetic algorithm with wavelet approximation. In: Proceedings of Fifth World Congress on Intelligent Control and Automation. 2004. DOI: 10.1109/EDM.2018.8435093

[14] Iwasaki NMM. Evolutionary identification algorithm for unknown structured mechatronics systems using GA. In: Proceedings of 2000 26th Annual Conference of the IEEE Industrial Electronics Society. 2000. DOI: 10.1109/IECON.2000.972388

[15] Wakizono M, Hatanaka T, Uosaki K. Time varying system identification with immune based evolutionary computation. In: Proceedings of 2006 SICE-ICASE International Joint

Conference. 2006. DOI: 10.1109/SICE.2006.315098

[16] Kawada K, Yamamoto T. Evolutionary identification using closed-loop data for a mechanical system. Proceedings of SICE Annual Conference 2010. 2010

[17] Ghiglino P, Forshaw JL, Lappas VJ. Online evolutionary swarm algorithm for selftuning unmanned flight control laws. Journal of Guidance, Control, and Dynamics. 2015;**38**(4):772-782

[18] Whorton M. Closed loop system identification with genetic algorithms.

[19] Jiang Q, Zhang J. Fuzzy multi-objective evolutionary algorithm based structure identification of polynomial systems. In: Proceedings of the 33rd Chinese Control Conference. July 2014. pp. 6571-6576

[20] Mondoloni S. A genetic algorithm for determining optimal flight trajectories. In: Guidance, Navigation, and Control Conference and Exhibit. 1998

[21] Murrieta-Mendoza A, Hamy A, Botez RM. Four- and three-dimensional aircraft reference trajectory optimization inspired by ant colony optimization. Journal of Aerospace Information Systems. 2017;**14**(11):597-616

[22] Cheng Y-F, Balakrishnan S. Suboptimal atmospheric trajectory design using genetic algorithms with variable mutation. In: 33rd Aerospace Sciences Meeting and Exhibit. Jan 1995

[23] Subbarao K, Shippey BM. Hybrid genetic algorithm collocation method for trajectory optimization. Journal of Guidance, Control, and Dynamics. 2009;**32**(4):1396-1403

[24] Kumar GN, Ahmed MS, Sarkar A, Talole S. Reentry trajectory optimization using gradient free algorithms. IFAC-PapersOnLine. 2018;**51**(1):650-655

[25] Evans B, Walton S. Aerodynamic optimization of a hypersonic reentry vehicle based on solution of the Boltzmann BGK equation and evolutionary optimization. Applied Mathematical Modeling. 2017;**52**:215-240. DOI: 10.1016/j.apm.2017.07.024

[26] Du D-Z, Pardalos PM, Wu W. History of Optimization. Boston, MA: Springer US; 2009. pp. 1538-1542. DOI: 10.1007/978-0-387-74759-0 268

[27] Holland J. Adaption in Natural and Artificial Systems. Ann Arbor, MI, United States: The University of Michigan Press; 01 Dec 1975:206. ISBN 10: 0472084607. ISBN 13: 9780472084609

[28] Nyew HM, Abdelkhalik O, Onder N. Structured-chromosome evolutionary algorithms for variable-size autonomous interplanetary trajectory planning optimization. Journal of Aerospace Information Systems. 2015;**12**(3):314-328

[29] Jia D, Vagners J. Parallel evolutionary algorithms for uav path planning. In: AIAA 1st Intelligent Systems Technical Conference. 2004

[30] Qu X, Duan H. Three dimensional trajectory planning of unmanned aerial vehicles based on quantum differential search. In: Proceedings of the 33rd Chinese Control Conference. 2014

[31] Anh HPH, Son NN, Nam NT. Adaptive evolutionary neural control of perturbed nonlinear serial pam robot. Neurocomputing. 2017;**267**:525-544

[32] Kargupta H. Gene expression: The missing link in evolutionary computation. 1997;**9**. Available from: https://www.osti.gov/biblio/524858-gene-expression-missing-link-evolutionary-computation

[33] Goldberg DE, Korb B, Deb K. Messy genetic algorithms: Motivation, analysis, and first results. In: Complex Systems. Vol. 5. Complex Systems Publications, Inc; 1989

[34] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95 – International Conference on Neural Networks. Available from: https://ieeexplore.ieee.org/document/488968/authors#authors

[35] Yang X-S. Firefly algorithm. In: Nature-Inspired Metaheuristic Algorithms. 2nd ed. Luniver Press; 2010

[36] Yang X-S, Deb S. Cuckoo search via levy flights. In: Proceedings of 2009 World Congress on Nature & Biologically Inspired Computing. 2009

[37] Karaboga D. An idea based on honey bee swarm for numerical optimization. In: Technical report TR06. Kayseri/Trkiye: Erciyes University, Engineering Faculty Computer Engineering Department; Oct 2005

[38] Dorigo M. Optimization, learning and natural algorithms [PhD dissertation]. Italy: Politecnico di Milano; 1992

[39] Biesbroek R. Study of genetic algorithm settings for trajectory optimisation. In: 54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law. 2003

[40] Goldberg DE. Sizing populations for serial and parallel genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms. Mar 1989. pp. 70-79, also TCGA Report 88004

[41] Harik G, Cant-Paz E, Goldberg DE, Miller BL. The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation. 1999;**7**(3):231-253

[42] Chen T, Tang K, Chen G, Yao X. A large population size can be unhelpful in evolutionary algorithms. Theoretical Computer Science. 2012;**436**:54-70

[43] Cage P, Kroo I, Braun R. Interplanetary trajectory optimization using a genetic algorithm. In: Astrodynamics Conference. 1994

[44] Wuerl A, Crain T, Braden E. Genetic algorithm and calculus of variations-based trajectory optimization technique. Journal of Spacecraft and Rockets

[45] Yokoyama N, Suzuki S. Modified genetic algorithm for constrained trajectory optimization. Journal of Guidance, Control, and Dynamics. 2005;**28**(1):139-144

[46] Yokoyama N, Suzuki S. Trajectory optimization via modified genetic algorithm. In: AIAA Guidance, Navigation, and Control Conference and Exhibit. 2003

[47] Silva PR, Abreu IS, Forte PA, Amaral HMCD. Genetic algorithms for satellite launcher attitude controller design. Inteligencia Artificial. 2019;**22**(63):150-161

[48] Chai R, Savvaris A, Tsourdos A, Xia Y, Chai S. Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm. IEEE Transactions on Cybernetics. 2018:114

[49] Tighzert L, Aguercif T, Fonlupt C, Mendil B. Intelligent trajectory planning and control of a humanoid robot using a new elitism based selfish gene algorithm. In: 2017 6th International Conference on Systems and Control (ICSC). May 2017

[50] Azarkaman M, Aghaabbasloo M, Salehi ME. Evaluating GA and PSO

evolutionary algorithms for humanoid walk pattern planning. In: 2014 22nd Iranian Conference on Electrical Engineering (ICEE). 2014

[51] Huang H, Zhou H, Cai Y. Study on multi-path planning and tracking control of the ucav based on evolutionary algorithm. In: 2015 15th International Conference on Control, Automation and Systems (ICCAS). Oct 2015

[52] Cooper MA, Heidlauf PT. Nonlinear feed forward control of a perturbed satellite using extended least squares adaptation and a luenberger observer. Journal of Aeronautics & Aerospace Engineering. Jan 2018;**07**(01)

[53] Sands T. Deterministic Artificial Intelligence. IntechOpen; 2019. ISBN: 978-1-78984-112-1

[54] Brownlee J. Swarm algorithms. Lulu.com, 2011. [Online]. Available from: http://www.CleverAlgorithms.com

[55] Pontani M, Conway BA. Particle swarm optimization applied to space trajectories. Journal of Guidance, Control, and Dynamics. Oct 2010;**33**(5)

[56] Akhmedova S, Stanovov V, Erokhin D, Semenkina O. Success-history based biology inspired algorithms for global trajectory optimization. IOP Conf. Series: Materials Science and Engineering. 2019;**537**. DOI: 10.1088/1757-899X/537/5/052008

[57] Liu C, Gao Z, Zhao W. A new path planning method based on firefly algorithm. In: 2012 Fifth International Joint Conference on Computational sciences and Optimization. 2012. DOI: 10.1109/CSO.2012.174

[58] Inghilterra G, Arrigoni S, Braghin F, Cheli F. Firefly algorithm-based nonlinear mpc trajectory planner for autonomous driving. In: 2018 International Conference of Electrical and Electronic Technologies for Automotive. 2018

[59] Raja NSM, Manic KS, Rajinikanth V. Firefly algorithm with various randomization parameters: An analysis. In: Proceedings of SEMCCO 2013, Part I, LNCS 829. Switzerland: Springer International Publishing; 2013. pp. 110-121

[60] Meena S, Chitra K. An approach of firefly algorithm with modified brightness for pid and i-pd controllers of siso systems. Journal of Ambient Intelligence and Humanized Computing. Berlin Heidelberg: Springer; Mar 2018. pp. 1–9. DOI: 10.1007/s12652-018-0747-x

[61] Mahapatra G, Mahapatra S, Banerjee S. A study of firefly algorithm and its application in non-linear dynamic systems. International Journal of Trend in Scientific Research and Development. 2018;**2**(2)

[62] Alao OJ, Metu MO, Amibor IP, Ibe CC, Oluyombo OW. Firefly optimization design and simulation of a single-axis helmholtz coils for spacecraft components testing. International Journal of Engineering and Applied Sciences. 2018;**5**(11)

[63] Shabbir F, Omenzetter P. Application of firefly algorithm to the dynamic model updating problem. 2015

[64] Corne D, Glover F, Dorigo M. New Ideas in Optimization, Advanced Topics in Computer Science Series. McGraw-Hill; 1999. Available from: https://books.google.com/books?id=-LYHngEACAAJ

[65] Dorigo M, Birattari M, Stutzle T. Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine. Nov 2006; **1556-603X/06**

[66] Ma Y-N, Gong Y-J, Xiao C-F, Gao Y, Zhang J. Path planning for autonomous underwater vehicles: An ant colony algorithm incorporating alarm pheromone. IEEE Transactions on Vehicular Technology. 2019;**68**(1): 141-154

[67] Ntagiou E. Earth observation and data relay constellation missions' planning with ant colony optimization [PhD dissertation]. University of Surrey; 2019

[68] Zhang T-J, Shen H-X, Shang X, Wang J, Li H-N. Optimal mission planning of leo debris collecting. In: Proceedings of 2018 SpaceOps Conference. 2018. DOI: 10.2514/6.2018-2412

[69] Wase V. High-thrust interplanetary spacecraft trajectory optimization using cuda [PhD dissertation]. Institutionen fr informationsteknologi: Uppsala Universitet; 2018

[70] de Oliveira Alves GF, Becceneri JC, Sandri S. A balancing heuristic for spacecraft equipment layout optimization. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZIEEE). Aug 2015. pp. 1-8

[71] Murrieta-Mendoza A, Botez RM, Bunel A. Four-dimensional aircraft en route optimization algorithm using the artificial bee colony. Journal of Aerospace Information Systems. 2018; **15**(6):307-334

[72] Zhong SA, Dong YF. Foundations and practical applications of cognitive systems and information processing. Advances in intelligent systems and computing. In: Artificial Bee Colony Algorithm for Parametric Optimization of Spacecraft Attitude Tracking Controller. Vol. 215. Berlin, Heidelberg: Springer; 2014

[73] Ding L, Wu H, Yao Y. Chaotic artificial bee colony algorithm for system identification of a small-scale unmanned helicopter. International Journal of Aerospace Engineering. 2015. DOI: 10.1155/2015/801874

[74] Banerjee S, Bharadwaj A, Gupta D, Panchal V. Remote sensing image classification using artificial bee colony algorithm. International Journal of Computer Science and Informatics. 2012;**2**(3). DOI: 10.1155/2015/801874

[75] Liu X, Li D, Dong N, Ip WH, Yung KL. Non-cooperative target detection of spacecraft objects based on artificial bee colony algorithm. IEEE Intelligent Systems. 2019:1-1