A simheuristic approach for evolving agent behaviour in the exploration for novel combat tactics

Chiou-Peng Lam
Edith Cowan University

Martin Masek
Edith Cowan University

Luke Kelly
Edith Cowan University

Michael Papasimeon

Lyndon Benke

Operations Research Perspectives 6 (2019) 100123

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

A simheuristic approach for evolving agent behaviour in the exploration for novel combat tactics

Chiou-Peng Lam[a], Martin Masek[*,a], Luke Kelly[a], Michael Papasimeon[b], Lyndon Benke[b]

[a] School of Science, Edith Cowan University, 270 Joondalup Drive, Joondalup, WA 6027, Australia
[b] Joint & Operations Analysis Division, Defence Science Technology Group, Department of Defence, 506 Lorimer Street, Fishermans Bend, VIC 3207, Australia

ARTICLE INFO

Keywords:
Simheuristics
Genetic algorithms
Multiagent simulations
Stochastic combinatorial optimization
Finite state machines

ABSTRACT

The automatic generation of behavioural models for intelligent agents in military simulation and experimentation remains a challenge. Genetic Algorithms are a global optimization approach which is suitable for addressing complex problems where locating the global optimum is a difficult task. Unlike traditional optimisation techniques such as hill-climbing or derivatives-based methods, Genetic Algorithms are robust for addressing highly multi-modal and discontinuous search landscapes. In this paper, we outline a simheuristic GA-based approach for automatic generation of finite state machine based behavioural models of intelligent agents, where the aim is the identification of novel combat tactics. Rather than evolving states, the proposed approach evolves a sequence of transitions. We also discuss workable starting points for the use of Genetic Algorithms for such scenarios, shedding some light on the associated design and implementation difficulties.

1. Introduction

Computational simulation is increasingly employed to support decision-making in real-world problems, from smart cities to military and transportation logistics. Simulations incorporating Computer Generated Forces (CGFs), play an increasingly vital role in military training, intelligence analysis and exploration of combat tactics. Typically, these simulations incorporate intelligent agents that capture individual and team decision-making, for a variety of reasons including the exploration and assessment of tactics for combat. CGFs (or agents)[1] with associated behavioural models typically take on roles such as an ally or an adversary. However, the development of these agent-based behavioural models in military simulation and experimentation remains a challenge [1]. Existing approaches can be classified into two main classes, namely manual and those incorporating some form of Artificial Intelligence (AI). The manual process of scripting these behavioural models is labour intensive, costly, requires domain expertise and may or may not involve the use of dedicated CGF behaviour authoring tools (e.g. Smart Bandits [2]). Commercial-Off-The-Shelf CGF (COTS-CGF) packages have commonly been used in the development of behavioural models for military training simulations. Many of these COTS-CGF incorporate some rudimentary form of AI and according to Toubman et al. [3], these packages do not support behaviour modelling through adaptive processes incorporating AI. They found even large COTS-CGF packages still rely on forms of scripting to drive agent behaviour, with some providing graphical interfaces to make it easier for users. They conclude that existing COTS-CGF packages have undesirably high complexity in terms of use and lacking AI techniques that would be beneficial for modelling agent behaviour and exploration of novel combat strategies.

Traditionally simulation and optimization were considered distinct approaches in operations research. The marriage between these two areas emerged around 2000 when the use of the term simulation optimization became more widespread and commercial discrete-event simulation software included some form of optimization modules instead of just statistical estimation modules. In recent years, the dichotomy between these two approaches is decreasing. With the exponential growth in computational power, it is now increasingly viable to do simulation optimization - simultaneously combining the exploration of details provided by simulations with the ability of optimisation techniques. The term Simheuristics was coined in 2015 [4] to represent a category of optimization algorithm that incorporates simulation to solve complex stochastic NP-hard combinatorial optimization problems. Juan et al. [5] have also argued that simheuristics be considered as a first-resort approach for solving real-world optimization problems; a class of problems typically characterised as being NP-hard, large-scaled, with a high degree of dynamism and uncertainty.

Simheuristics incorporating metaheuristics such as Simulated Annealing and Genetic Algorithms (GA) have increasingly been used to

* Corresponding author.
E-mail address: m.masek@ecu.edu.au (M. Masek).
[1] The terms agent and CGF are used interchangeably.

https://doi.org/10.1016/j.orp.2019.100123
Received 20 March 2019; Received in revised form 14 August 2019; Accepted 11 September 2019
Available online 12 September 2019

solve optimisation problems in the area of operations research. While traditional simulated annealing explores the solution space sequentially, GA operates on a population of individuals (solutions) where it uses a survival of the fittest principle to guide its search for the near-optimal solution. The applications of GA to solve optimisation problems include: task scheduling [6], Travelling Salesman [7], Portfolio selection [8], timetabling [9], and behavior modeling for air combat simulations [10,11]. GA is a global optimization approach, most applicable for complex non-linear models where locating the global optimum is a difficult task. Unlike traditional optimisation techniques such as hill-climbing or derivative-based methods, this technique is robust and suitable for addressing highly multi-modal and discontinuous search landscapes [12]. It should also be noted that there is a body of work that attempts to introduce the benefits of a population-based search to standard metaheuristic algorithms. For example, Shaabani and Kamalabadi [13] introduced a population-based version of simulated annealing applied to an inventory routing problem. In another work, Panadero et al. [14] introduce an agent-based framework, where a population of heterogenous search algorithm agents with shared memory can traverse a search space in parallel.

Optimisation algorithms found in commercial simulation software are mostly based on metaheuristics and predominantly GA [15,16]. Figueira et al.[17] also stated that optimization modules in discrete-event simulation software are predominantly metaheuristic methods owing to the flexibility of these techniques to tackle different types of search space and achieve good (near-optimal) solutions. A point to note is that optimization modules for commercial simulation software are black box in nature as exemplified by the GA-based methods both in SimRunner and in Siemen's PLM. A survey of research papers associated with simheuristics also showed that GA is a commonly used optimization method [18,19]. Similar to many other optimisation methods, GA-based approaches require a number of decisions to be made as their efficiency depends on many parameters, such as representation of the solution, the initial population, selection strategy, and recombination operators. The choice of a good combination of values for these parameters has a significant impact on the performance of the GA. Owing to the large number of combinatorial possibilities [20,21], finding a suitable set of values for these parameters is a challenge. However, in most existing work, a discussion is typically presented about the importance of determining appropriate parameters values and then followed by a sets of values employed in the study. Some guidance or rationale from an implementation perspective would be helpful to shed some light on the design and implementation difficulties associated with using GA and promote a wider discussion and uptake of metaheuristics in simulation-optimization. Using feedback from simulation can guide the search for better solutions as well as a better understanding of the system overall.

This paper considers the use of GA for evolving agent-based behaviour models associated with novel combat tactics. This problem is one involving a co-adaptive environment where the individual CFG and the others are one part of a dynamic environment (e.g. simulation). The formulation of fitness function for assessing behaviour models via some form of mathematical equations is challenging and in some instances, impossible. To address this, our approach is simheuristics, with the GA interacting with an environment (simulation) which provides utility-related feedback (in the form of a measurement of performance related to some time-varying state of the environment). In this paper we first examine GA concepts, design and experimental considerations in terms of their applications for automatic generation of agent-based behavioural models, specifically in terms of manoeuver optimization where the aim is the exploration and identification of novel combat tactics. A within visual range combat manoeuver, the Stern Conversion as outlined in Shaw [22], will be used here as the running example tactical manoeuver to illustrate considerations associated with the use this evolutionary approach. The contribution of this paper is two fold; namely we outlined a simheuristics GA-based approach for exploration of novel combat tactics associated with CGFs and we identify design considerations and implementation decisions as well as share insights where a researcher starting to work with GA in simheuristics may find such information constructive.

The following section describes associated preliminaries, namely components of GA and Section 3 outlines the Stern Conversion manoeuver and its corresponding behaviour model (in the form of finite state machine (DSTG-AT)). Section 4 discusses concepts and steps associated with our GA-based approach as well as GA-based design considerations for exploration of novel combat tactics associated with CGFs. Lastly, discussion and conclusion are found in Section 5.

2. Genetic algorithms (GA)

The GA is a general purpose search algorithm which was introduced by Holland [23]. It is biologically inspired and is based on Darwin's theory of survival of the fittest. Basically, the approach involves maintaining a population of individuals (chromosomes) that represent candidate solutions to the actual problem. The population evolves over time through a competitive process and controlled variations. Although there are different variations of GA, they all have a number of common elements: representation, fitness function and population management. The GA terminates when the stopping criteria is satisfied.

2.1. Representation

A problem-specific component which is also a key component in the development of a GA-based approach is the representation or encoding of the solution for the specific problem. With direct manipulations of the encoding by the GA-based approach, a specific encoding schema can severely limit the universe by which the GA based approach perceives its world. Traditionally, fixed-length and binary string encodings have been the dominating encoding; with sequences of 0's and 1's where the position of the digit correlates to some aspect of the solution. Instead of binary strings, a similar approach that supports greater precision and complexity involved integers or real number encoding [24]. Specifically, a solution or chromosome is a vector of real numbers in a real-coded GA. However a point to note here is that it is also possible to have encodings that are not fixed-length and fixed-order binary encoding. These include many-value encoding [25], messy genetic algorithms [26] and delta encoding [27].

2.2. Fitness function

The second problem-specific component is associated with evaluation of the goodness of an individual with respect to the problem under consideration. The goodness of a chromosome can either be represented by some function that will be optimised or is some form of a performance measurement obtained via the interaction of the GA with some time-varying state of the environment. The terms, Fitness function and Objective function, have often been used synonymously. The objective function is a type of phenotype, a characteristic associated with the specific problem. The fitness function is a monotonic function of the objective function and determines probabilistically how the individual will be subsequently used (discarded or reproduced) in the algorithm. Better solutions for the problem under consideration are given higher fitness values. It is vital that the fitness function is designed appropriately; otherwise the GA will either have great difficulty in converging or it will converge to an inappropriate solution. The design of the appropriate fitness function is often not a straightforward task and in many instances, this task may be achieved using human evaluation (interactive GA) or using simulations.

2.3. Population management

GAs work with a population of candidate solutions (chromosomes) encoded in some manner. The components of the chromosome are known as genes and the possible values associated with each gene are called alleles (e.g. genes in a binary string have 2 alleles: 0 and 1). GAs start with an initial population which is usually randomly generated [28]. This population is evolved iteratively until some stopping criterion is met, whereby the final population will contain solutions with high fitness. At the core of the GA process, are crossover and mutation operations that works on individuals in a population(t) to create a population($t + 1$) with an increased overall fitness. Commonly, a GA works on a static size population and three operators are at work in each generation to generate population($t + 1$) from population(t). These three genetic operators are selection, crossover and mutation. They play an important role; the selection and reproduction process support exploitation where the GA can conduct local search of promising regions and the crossover and mutation operators support exploration of untested regions of the solution space. The crossover operation produces two offspring by combining parts from a pair of parents [29] and the mutation operator changes a gene based on the mutation probability, typically a very small number [30,31].

There are two different population management models, namely generational or steady-state. In the generational model, the set of parents in each generation is totally replaced by the offspring whereas in the steady-state model, p members of the population are replaced by p offspring in each consecutive generation. Elitist selection guaranteed that the fittest individuals are included in the next generation (either the best or a few of the best individuals). In terms of the selection methods used to select parents for reproduction, these include: Stochastic Universal Sampling, Roulette-wheel selection, Tournament selection and Rank selection [32].

3. Running example: tactical manoeuver: stern conversion

To have a better appreciation for the discussions and the results presented here, we believe that some basic understanding of the employed aircraft combat manoeuver is required. The Stern Conversion [22] is a one-vs-one within visual range combat scenario where two aircraft are within visual range and initially flying towards each other (Fig. 2: time step a).The aim of the blue aircraft (Interceptor) is to achieve a position where it is behind the red (Target) aircraft and following it. In this position, the blue aircraft can maintain a sensor lock on red, while being safe from red's observation.

In executing this manoeuvre, the blue aircraft will first turn away from the red aircraft in time step b to achieve the Required Displacement between the two aircraft. Next, the blue aircraft will continue to fly parallel to red until it is within Conversion Range (Fig. 2: time step c). Lastly, the blue aircraft turns to position itself behind the red aircraft (Fig. 2: time step d).

4. A simheuristic GA-based approach and design considerations for exploration of novel combat tactics

We examine GA concepts, design and experimental considerations in terms of its applications for automatic generation of agent-based behavioural models, specifically in terms of manoeuver optimization where the aim is the exploration and identification of novel combat manoeuvres. Specifically we use the Stern Conversion manoeuvre as a reference point to illustrate design considerations and for discussions. A researcher wanting to develop a simheurisitc GA-based approach for the exploration and identification of novel combat manoeuvres for CGFs would need to carry out the following:

1. First, we need to consider appropriate models that can capture the behaviour of CGFs for specific combat manoeuvres. At this point we also need to identify a suitable combat simulator that supports the interactions between the Blue and Red agent.
2. Second, from the perspectives of the GA-based approach, we need to examine the way in which the combat manoeuvre has been modelled and how it fits into the GA framework, definition of solution space, solution representation as chromosomes and different measures of effectiveness and their contributions in the formulation of the fitness function.
3. Third, we need to decide on the various parameters of the GA-based approach, including those associated with population management and termination conditions.
4. Evaluation considerations given the stochastic nature of a GA-based approach.

4.1. Step 1: agent-based behaviour model for stern conversion

Typically, models capturing behaviour associated with CGFs are in the form of if-then-else rules which have been manually scripted. However Finite State Machines (FSM) and behaviour trees have also been used in work involving the use of machine learning for automatic generation of behaviour models of CGFs. The behaviour model for Stern Conversion used here is a finite state machine and is described in the following section.

4.2. DSTG-AT: DSTG agent incorporating kinematic knowledge for the stern conversion manoeuvre

The Australian Defence Science and Technology Group (DST Group) have developed an air-to-air combat simulator, ACE Zero [33] and an aircraft controller agent in the form of FSM for enacting the above Stern Conversion manoeuvre. Each state in this FSM corresponds to a particular section of the manoeuvre, with transitions occurring once the aircraft achieves the desired position and orientation as defined by Shaw (1985). A description of the states from this FSM is provided in Table 1. Whilst in each of these states, the blue aircraft also tries to match the altitude of the red aircraft.

Previously, we have proposed a generic FSM model for which transition can be evolved [11]. The FSM we use here is one instantiation of this generic model, shown in Fig. 3. This is a modified version of the DSTG aircraft controller FSM, which we refer to as DSTG-AT (All Transitions), where we have kept the original state actions from the DST model but have removed the prescriptive set of transitions, instead making transitions between all states possible. Transition conditions of DSTG-AT are determined using the kinematic properties of the two aircraft (Blue and Red), namely, the position (**p**), velocity (**v**) and acceleration (**a**) along three Cartesian components (i.e. the x, y and z axis). These kinematic properties are are easy to obtain from the simulation environment that is used to calculate the fitness, as they form an integral part of the simulation. When seeking to instantiate an optimized solution in real combat, the velocity (and acceleration) of the opponent can be estimated by taking differences (and differences of differences) of observed position. These kinematic parameters are calculated using the relative components of red aircraft to blue and are defined below.

- Relative Distance as:

$$\mathbf{\Delta p} = (\Delta p_x, \Delta p_y, \Delta p_z) = (p_{xRed} - p_{xBlue}, p_{yRed} - p_{yBlue}, p_{zRed} - p_{zBlue})$$

Table 1
States and transitions for the FSM stern conversion agent.

State	Action
(a) Pure pursuit	Point the aircraft at and fly directly towards the red aircraft.
(b) Fly relative bearing	Turn by **turn angle** and fly straight on that bearing.
(c) Flying offset	Fly parallel to red
(d) Converting	Turn to match red's heading whilst remaining behind it, not coming closer than **no closer range.**

Table 2
An example of the transition logic that must all be satisfied for a change of state from FlyingOffset to FlyRelativeBearing. The transition parameters (A to R) act as thresholds for the relative position ($\mathbf{\Delta p}$), velocity ($\mathbf{\Delta v}$) and acceleration ($\mathbf{\Delta a}$) in each transition.

$A_{ij} < \Delta p_x < B_{ij}$	$G_{ij} < \Delta v_x < H_{ij}$	$M_{ij} < \Delta a_x < N_{ij}$
$C_{ij} < \Delta p_y < D_{ij}$	$I_{ij} < \Delta v_y < J_{ij}$	$O_{ij} < \Delta a_y < P_{ij}$
$E_{ij} < \Delta p_z < F_{ij}$	$K_{ij} < \Delta v_z < L_{ij}$	$Q_{ij} < \Delta a_z < R_{ij}$

- Relative Velocity as:

$$\mathbf{\Delta v} = (\Delta v_x, \Delta v_y, \Delta v_z) = (v_{xRed} - v_{xBlue}, v_{yRed} - v_{yBlue}, v_{zRed} - v_{zBlue})$$

- And Relative Acceleration as:

$$\mathbf{\Delta a} = (\Delta a_x, \Delta a_y, \Delta a_z) = (a_{xRed} - a_{xBlue}, a_{yRed} - a_{yBlue}, a_{zRed} - a_{zBlue})$$

This transformation to relative values allows us to consider behaviour of the red aircraft in terms of where it is and where it is going relative to the blue aircraft, how the blue and red aircraft are increasing or decreasing their separation and bearing.

To determine if a transition of a specific state is to be triggered, each possible transition from that specific state is checked whether the Relative Position, Velocity and Acceleration (separately for each of their Cartesian components) is within a respective minimum and maximum bound for that transition to occur. Evaluation for the transition between these two states, i and j, are made against 18 constants (denoted by $A_{ij}, B_{ij}, C_{ij}, D_{ij} \cdots R_{ij}$). As an example, Table 2 shows the conditions that must all be satisfied for a transition from FlyingOffset to FlyRelativeBearing.

Since it is possible to go from each state to three other states, there are $3 \times 18 = 54$ parameters to check within each state. As there are 4 states in DSTG-AT, the total number of parameters is 216. The parameters for each of the 12 transitions between states are listed in Table 3.

If the conditions for only one transition from the FSMs current state are met, that transition is taken. If conditions for more than one transition are met, one of these transitions is taken at random. The following original Tactical Parameters for DST Agent are still utilised in DSTG-AT:

- TURN_ANGLE for the FlyRelativeBearing state, set to the default 15.0°.
- NO_CLOSER_RANGE for the Converting state, set to the default 2.0 nautical miles.

4.3. Step 2: GA considerations in practice

The first consideration in the development of a GA-based approach is the representation or encoding of the solution for the specific problem. This is illustrated in the following section using our running example, the Stern Conversion manoeuvre and its associated behaviour model DSTG-AT. The generic steps associated with a GA-based approach are shown in Fig. 1. Chromosome encoding is associated with the step Generate Initial Population.

Table 3
The transition parameters for the Stern Conversion FSM (DSTG-AT).

		Current state			
		Pure pursuit	Fly relative bearing	Flying offset	Converting
Next state	Pure pursuit		$A_{21} \cdots R_{21}$	$A_{31} \cdots R_{31}$	$A_{41} \cdots R_{41}$
	Fly relative bearing	$A_{12} \cdots R_{12}$		$A_{32} \cdots R_{32}$	$A_{42} \cdots R_{42}$
	Flying offset	$A_{13} \cdots R_{13}$	$A_{23} \cdots R_{23}$		$A_{43} \cdots R_{43}$
	Converting	$A_{31} \cdots R_{31}$	$A_{24} \cdots R_{24}$	$A_{34} \cdots R_{34}$	

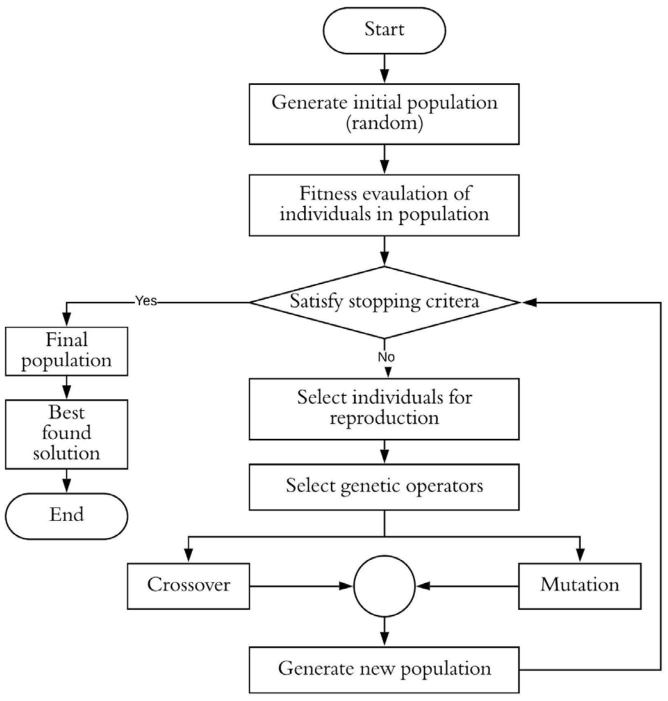

Fig. 1. Steps in a basic genetic algorithm.

4.4. Chromosome encoding

The aim is to evolve a sequence of transitions between states in DSTG-AT such that a stern conversion manoeuvre is successfully executed by the Blue Agent (DSTG-AT) against a Red Agent. Thus the chromosome represents the set of all possible transitions in DSTG-AT, each transition with its respective conditions that must be met for the transition to be triggered. The optimal solution encodes the transitions and their respective conditions which culminates in the Blue agent executing a successful stern conversion against the Red agent.

Fig. 4 shows the proposed chromosome representation. Each gene is a real number in the range of [0.0–1.0]. Table 2 provides the details for these genes, for example, gene A and gene B is the minimum and maximum bound associated with Δp_x respectively. These gene values are associated with the size of a simulated environment. The length of this chromosome is 216, representing the bounds associated with each of the 12 transitions in DSTG-AT.

4.5. Fitness evaluation of individuals in population

Fitness evaluation is the step following population initialisation in a GA (ref. Fig. 1). The evaluation method employed to assess the goodness of individual solution is another critical design consideration in a GA-based approach. For the optimal solution to be meaningful, it is vital to develop functional performance metric that captures the effectiveness of strategies used by the individual (e.g. Blue agent) against its opponents (Red agent). In the case of learning tactical combat strategies involving stochastic-based behaviour models it is computationally infeasible to objectively assess the quality of a strategy using some form of mathematical equation; though it is relatively easy to let the respective Agents (Blue vs Red) interact via an air-to-air combat simulator. The fitness of the chromosome (Blue agent) is then a

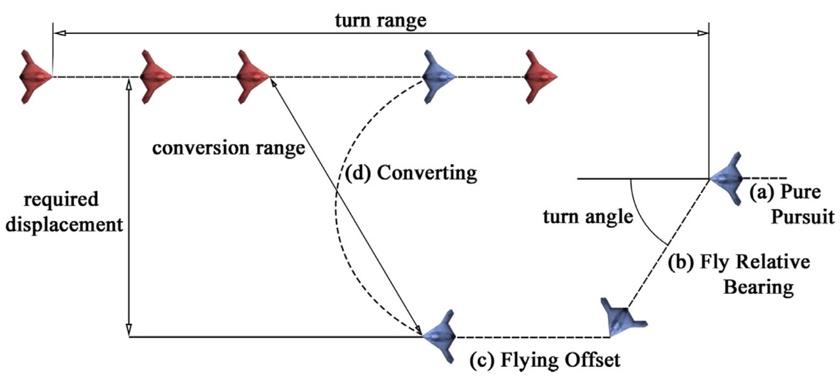

Fig. 2. Blue Interceptor performing a Stern Conversion on Red Target. Aircraft positions at time steps 1–4 shown. Reproduced from Shaw [22]. The figure also shows the set of states (a–d) and tactical parameters used in the finite state machine model (DSTG-AT) for enacting the Stern Conversion manoeuvre (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

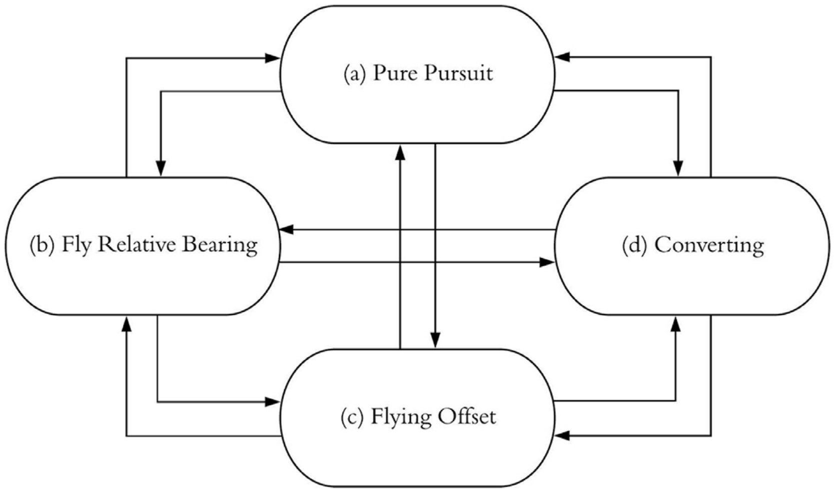

Fig. 3. DSTG-AT: modified DSTG finite state machine that implemented the stern conversion manoeuvre.

performance measurement from the interactions of the Blue agent against the Red agent in terms of successfully completing the specific task, in this instance, using some measures of success for stern conversion. In our specific example, the air-to-air combat simulator used is Ace Zero and the specific tasks here are: (1) determining a set success measure and (2) a method to use the success measure to calculate the fitness value of an individual.

4.6. Measure of stern conversion success

To measure stern conversion success of the blue agent (DSTG-AT) we consider the blue aircraft to have succeeded if during the simulation run the following criteria have been met:

1. The range (distance) between the two aircraft is between 500 feet and 3000 feet.
2. The target aircraft (Red) is within 30° of the attacking aircraft's (Blue) nose.
3. The attacking aircraft (Blue) is within 30° of the threat aircraft's (Red) tail.
4. The separation in altitude between the two aircraft is less than 500 feet.
5. The difference in velocity is less than 100 knots.

4.7. Calculation of fitness of a chromosome

The objective is to maximize the fitness score for the chromosome. To calculate the fitness, the individual chromosome being evaluated is used to instantiate the blue agent in Ace-Zero. Similarly, the red agent is instantiated with values for the specific agent being employed as the adversary. The Ace-Zero simulation is then run for 250 s. The fitness of a blue agent is initialised to 0 at the start of the simulation. During the simulation, each of the five conditions listed above are checked at intervals of 1 s (in terms of simulated time). In addition, to provide a less specific fitness factor, a sixth condition was introduced, that the attacker (Blue) is behind the opponent (Red). Though this condition is subsumed by a combination of conditions 2 and 3, it helps guide the evolution during the early phases towards more viable solutions. For each condition that is true at the particular point in time we add 1 to the fitness score. Thus, the more conditions that are true at a particular

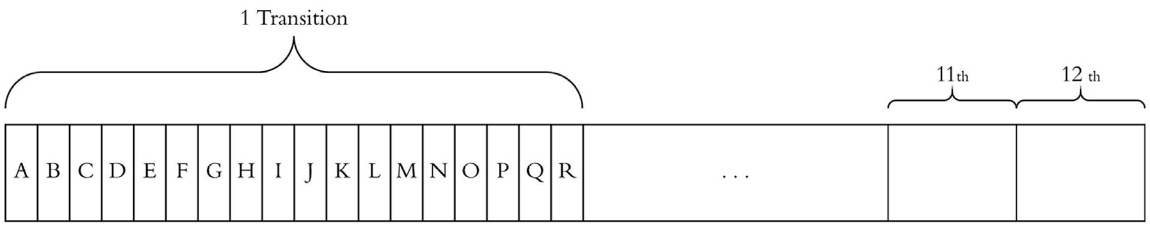

Fig. 4. Chromosome representation for evolving FSM transitions.

time, the higher the fitness for that time interval. Fitness score is summed across time intervals, so that a higher fitness results the longer a condition is true.

A point to consider in the calculation of the fitness score is the stochastic nature of the underlying behaviour model, where if the conditions for more than one transition to occur are met, one of the transitions is taken at random. This means that subsequent simulation runs of the same scenario with the same agent models may result in a different outcome. To demonstrate this, we set up the following:

- **Sim 1:** 50 evaluations, each consisting of calculation of fitness value using 1, 5 and 10 simulation runs involving an identical Blue Agent (DSTG-AT) versus Red Agent flying in a straight line respectively
- **Sim 2:** 50 evaluations, each consisting of calculation of fitness value using 1, 5 and 10 simulation runs involving an identical Blue Agent (DSTG-AT) versus Red Agent being the default stern conversion agent

Note that each evaluation run uses the same initial Blue and Red agent and we examine the fitness score of the Blue agent over 50 evaluations.

Figs. 5 and 6 shows the results for Sim 1 and Sim 2 respectively, where the fitness score associated with a single simulation (blue line) showed quite a bit of variability across the 50 simulations. This means that we might lose a potentially good solution as the specific individual did not perform well by chance or we might take a poor solution into future generations as it performed really well by chance. To address this sort of situation, one approach is to calculate each fitness score of an individual as an average performance from a repeated number of simulation runs using the same individual. The orange plot in both figures shows fitness value plots calculated using the average of 5 simulations per run and the green colour plot where fitness value was calculated from an average fitness of 10 simulations per run. This showed the fitness score of the same individual having less variability when calculated as an average across 5 and 10 repeats of the simulation respectively. Calculating the fitness values as an average of a number of simulation runs increases the computation time for the approach and determining how many to use is clearly a trade-off between obtaining a more reliable value for fitness of an individual and the computation resources. From the set of empirical analysis carried out, it can be seen that the variances between using 5 or 10 is not very big, thus a suitable measure of fitness can be calculated as an average of 5 runs.

4.8. Step 3: genetic operators and parameter tuning

The discussion in the following sections pertains to the following

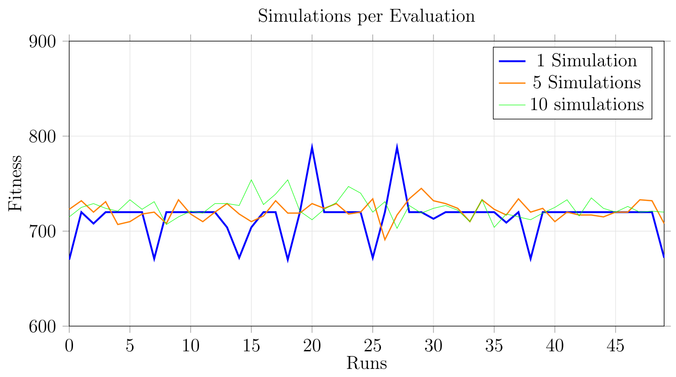

Fig. 5. Plot of fitness values from 50 evaluation runs, each consisting of calculation of fitness value using 1, 5 and 10 simulation runs respectively. The Blue Agent uses a DSTG-AT model against a Red Agent flying in a straight line (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

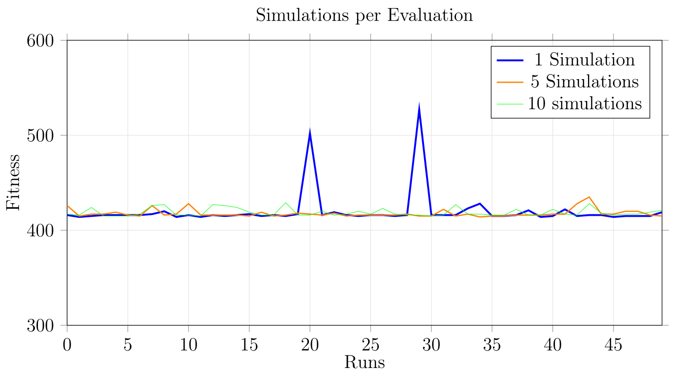

Fig. 6. Plot of fitness values from 50 evaluations, each consisting of calculation of fitness value using 1, 5 and 10 simulation runs respectively. The Blue Agent using a DSTG-AT model against a Red Agent is the default stern conversion agent (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

steps of Fig. 1: Select individuals for reproduction, Select genetic operators, Crossover and Mutation.

Parameter tuning is considered here to be a specific case of algorithm design as it normally includes all decisions associated with its implementation. The efficiency of a GA-based approach is very much dependent on carefully choosing values for its various parameters. These include parameters associated with the use of crossover, mutation, type of selection and replacement strategies associated with population management as well as the termination conditions. Parameter tuning is defined as finding the best values for a set of parameters for running a GA. There are two typical approaches: use one of the Standard parameter settings or customise the parameter settings for your specific problem. While historical parameter settings from the literature may not be the most optimal solution, they are a good starting point. Table 4 shows common recommended Standard parameter settings obtained from the literature and these include those of Dejong [34], Grefenstette [20] Goldberg [35] and Alander [36].

Using Dejong's values for population size, crossover and mutation probability in conjunction with the remaining parameters in Table 6, we apply our GA-based approach to the running example where the Blue Agent (DSTG-AT) is attempting to complete a stern conversion against a Red Agent flying in a straight line (Sim 1). The best fitness individual at the end of 300 generations has a fitness score of 925. Fig. 7 shows the fitness plot of the best individual and the average fitness of the population across 300 generations. From these plots, it can be seen that there was a substantial amount of time associated with exploration and exploitation before the plot plateaued out with small oscillations. The evolution process has not completely converged as evident from the oscillations associated with the two plots toward the 300th generation. Further improvement may be obtained by running the GA longer. The resulting trajectory plot of the best individual at the end of 300 generations is shown in Fig. 8, demonstrating the Blue agent completing a stern conversion against the red agent. For completeness, the trajectory plot of the best individual using the same evolution parameters but evolved against the stern conversion agent is shown in Fig. 9. Here the blue aircraft also positions itself behind and following the red, successfully completing a stern conversion. From these trajectories, it can be seen that a reasonable result is achievable using standard parameter

Table 4
Standard parameter settings from the literature.

	Dejong	Grefensette	Goldberg	Alander
Population size	50	30	30	50
Crossover probability P_c	0.6	0.9	0.6	0.5
Mutation probability P_m	0.001	0.01	0.033	0.002

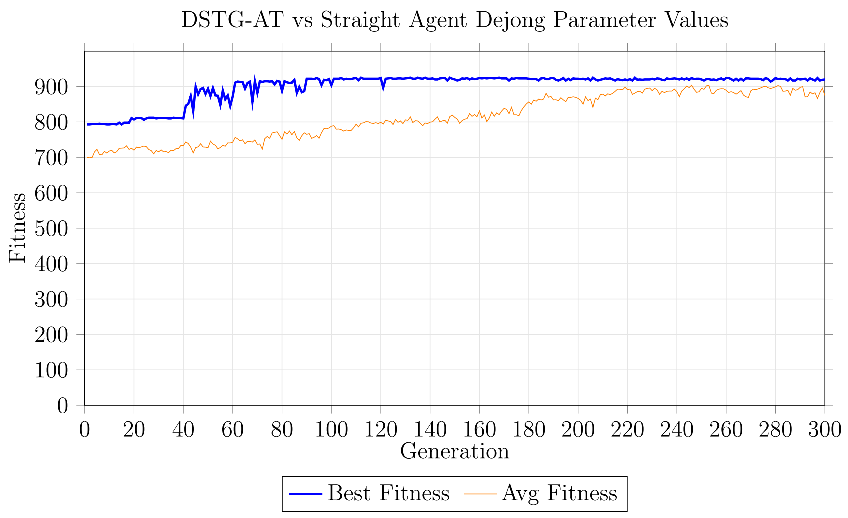

Fig. 7. Fitness plot associated with Dejong's parameter values for population size, crossover and mutation probability. Table 6 lists the remaining parameters used.

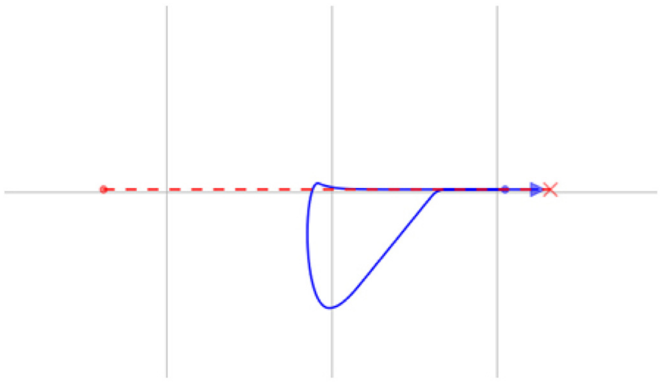

Fig. 8. Trajectory of the Blue Agent vs Red Agent from Sim 1 with the highest fitness upon termination of the GA using Dejong's parameter values for population size, crossover and mutation probability. The figure shows the initial starting positions of the two aircraft as dots, with the dashed line representing the trajectory of the red aircraft, terminating at the X symbol and the full line showing the trajectory of the blue aircraft, terminating at the arrow symbol. The figure shows a successful stern conversion has been made by the blue aircraft. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

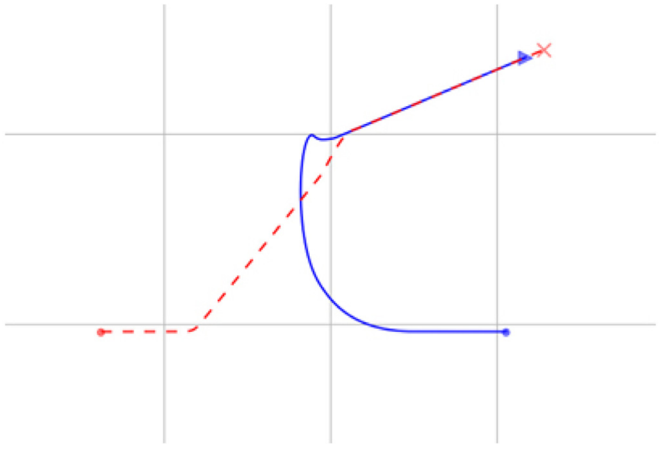

Fig. 9. Trajectory of the Blue Agent vs Red Agent from Sim 2 with the highest fitness upon termination of the GA using Dejong's parameter values for population size, crossover and mutation probability. This shows a successful stern conversion by the blue aircraft, which achieves a position behind and following the red aircraft. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

settings from the literature.

Customising the parameter settings could be labour intensive as it involves using heuristics from the literature and running many empirical experiments to systematically work out a set of appropriate values for the specific problem. Table 5 shows potential parameters and some of their associated options that need to be considered in the customisation process. Given that the set of parameters usually consists of a combination of real and integer values, exhaustive enumeration of the parameter space in the customisation process would not be possible. In practice, the approach taken in most studies is limited, usually using a fixed representation scheme and a chosen set of genetic operators associated with selection, crossover and mutation. Customisation typically involves only three parameters, namely population size, crossover probability and mutation probability. While some studies have explored approaches for parameter tuning, for example, in population sizing [37–39], the most common approach is the empirical method whereby customisation involves trying various values (in step-wise increments) for a specific parameter and the one that produces the best results is then used subsequently. Examining historical parameter settings from the literature again can serve as a guide to researchers as to possible initial values to start the customisation process.

Table 6 shows the chosen set of genetic operators associated with selection, crossover and mutation as well as the different values associated with population size, crossover probability and mutation probability which we will use to demonstrate parameter tuning of these three quantitative parameters. The first of the three parameters that we will discuss in terms of customisation is population size and its associated trade-offs; where choosing a small population may result in failing to adequately cover the solution space versus a large population resulting in a heavy computation load as well as a slow progression towards the optimal solution. The general consensus is that the more complex the problem, the larger the population size with some studies stating that it should be at least the equal to the length of the chromosome [40]. Another example of a Rule of Thumb for determining population size in the literature that is based on the work of Goldberg Deb & Clark [37] is given by Carroll [41], depending on the cardinality of the chromosome encoding, length of the chromosome and the number of combinations of gene values for the chromosome (schema size).

The second parameter typically tuned in the customisation process is the mutation probability (P_m). Mutation is a way of introducing variations associated with the value of the genes and promoting

Table 5
Summarizes different types of parameters for GA.

Quantitative parameters	
	• Population size • Crossover probability • Mutation probability • Offspring population size
Qualitative parameters	**Examples of associated options**
Crossover (Recombination)	• Single point crossover • Two point crossover • Arithmetic crossover • Uniform crossover
Mutation	• Gaussian • Bit-flip • Cauchy • Uniform
Parent selection	• Fitness proportionate: · Roulette-wheel · Stochastic universal sampling • Rank-based: · Linear ranking · Tournament selection
Survivor strategy	• Generational • Replacement (Steady-state) • Elitist
Representation	• Binary encoding • Gray encoding • Value encoding • Permutation encoding

diversity into the population. Examples of different mutation operator are shown in Table 5. The type of mutation operator used is usually problem/representation dependent. A change in gene value will occur with a probability of P_m and no change will be introduced with a probability of 1 - P_m. P_m is normally assigned a small value so that the mutation operation introduces small changes to the existing solutions without destroying good solutions. P_m is typically a very small value with some studies calculating its value as 1/l, where l is the length of the chromosome. A mutation probability of 1 would imply the whole chromosome undergoes mutation leading to a random search. Similar to population size, P_m can also be determined empirically.

The third and last parameter considered in the customisation process is the crossover probability (P_c). Crossover is a genetic operator that reproduce offspring by recombining building blocks from selected parents. Types of crossover operators are shown in Table 5. In One Point Crossover, a random crossover point is selected and the tails of the two selected parents are exchanged to produce two new off-spring. This operator can be generalised to a multipoint (e.g. 2) operator, where q randomly generated crossover points are used. The crossover operator is dependent on the representation of the individual (chromosome), otherwise invalid offspring may be produced. For instance in the case of a Travelling Salesman Problem with its representation being a graph, if care is not taken some offspring will be invalid – a result from crossover operations that produced offspring with some duplicated/missing cities. A crossover probability, P_c, defines how often the crossover operator is applied. Examining the Standard parameter settings, it can be seen that there is no single commonly acceptable value. Dejong [34] suggested a value of 0.6, Grefenstette [20] suggested 0.95 and Alander [36] proposed a value of 0.5. Similar to population size, P_c can also be determined empirically.

In the following section, we discuss some salient points associated with tuning the three parameters discussed above using our running example. In our empirical approach to tune the three parameters, we will use 3 values (50, 100 and 200) for population size, 4 values for crossover (P_c = 0.2, 0.4, 0.6, 0.8) and 4 values for mutation probability (P_m = 0.001, 0.01, 0.1, 0.2); incorporating some values from the literature (values shown in bold – being taken from Dejong (1990)) as part of the tuning process. It can be seen that to fully tune the three parameters empirically would be a combinatorial exercise, requiring 48 (3 × 4 × 4) experiments. The set of remaining parameters used is outlined in Table 6. The scenario used here involved the Blue Agent (DSTG-AT) attempting to complete a stern conversion against a Red Agent flying in a straight line. Figs. 10 and 11 are associated with empirical experiments for tuning for population size.

Fig. 10 shows the fitness plot for the best individual and the average fitness of the population across the 300 generations for an initial population size of 200. It found a very good individual early and that individual dominated until approximately generation 140th before a better individual was obtained. At this stage, it is showing that the exploration of the search space is rather limited as a high fitness individual from early on in the evolution process dominated for more than 100 generations. Fig. 11 (a)–(c) compare the trajectory associated with the manoeuvres for the highest fitness Blue agent (Blue line) upon termination of the GA process for initial population size of 50, 100 and 200 respectively. The red agent is flying in a straight line. The highest fitness value associated with the Blue agent is 913, 928 and 941 for population size of 50, 100 and 200 respectively. It can be seen that the trajectory of the Blue agent associated with initial population size of 200 is smoother that the other two. From these runs, it can be seen that an initial population size of 200 should be ultimately used.

Fig. 12 shows an instance associated with tuning mutation rate for our running example, evolving against the straight-line agent. This specific plot is associated with the following parameter values: population size = 50, P_c = 0.6, and the remaining parameters as shown in Table 6. P_m is varied, taking values from 0.001, 0.01. 0.1 or 0.2. Notice the plot associated with P_m = 0.001, the exploration process is taking much longer, with a lot more oscillations in the fitness values between generations early on in the evolution process as well as finally obtaining the individual with highest fitness value from the four different mutation rates. Notice the plots with mutation probability of 0.1 and 0.2 plateaued very early with a much lower fitness value for its best individual.

Fig. 13 shows an instance associated with tuning crossover probability for our running example, evolving against the straight-line agent. This specific plot is associated with the following parameter

Table 6
Set of genetic operators and associated parameters values involved in the customisation of population size, P_c and P_m.

Parameter	Value
Population Size	P = 50, 100 or 200
Maximum Generations	300
Simulations per individual	5
Selection operator	Stochastic universal sampling
Crossover operator	One point crossover
Crossover probability (P_c)	0.2, 0.4, 0.6 or 0.8
Mutation operator	Gaussian mutation
Mutation probability (P_m)	0.001, 0.01, 0.1 or 0.2
Elitism	Single best carried over
Termination	If the best chromosome fitness minus the average chromosome fitness is less than 10^{-3} for 10 consecutive runs OR if the chromosome best fitness doesn't change for 100 generations OR if maximum generations has been reached

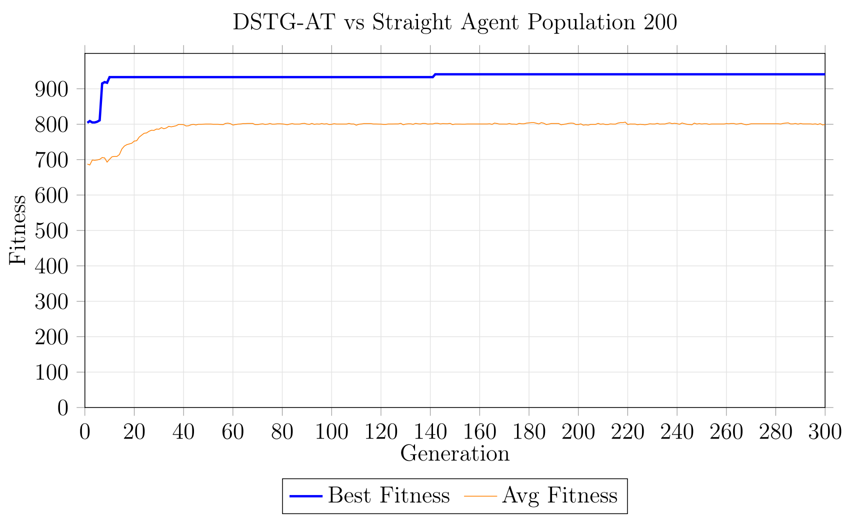

Fig. 10. Fitness plot associated with tuning population size: parameter settings involved: Population size = 200, P_c = 0.2, and P_m = 0.01. Table 6 lists the remaining parameters used.

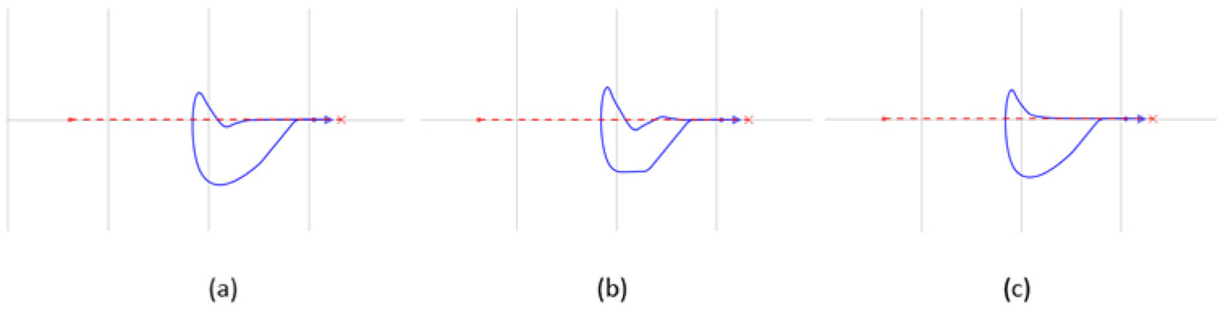

Fig. 11. Tuning Population size: trajectories of the Blue Agent with the highest fitness upon termination of the GA where the population size is 50 (a), 100 (b) and 200 (c). P_c = 0.2, and P_m = 0.01. Table 6 lists the remaining parameters used (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

values: population size = 50, P_m = 0.001, and the remaining parameters as shown in Table 6. P_c takes on a value from the following: 0.2, 0.4, 0.6 or 0.8. Notice the plot associated with P_c = 0.8, the plot for the fitness value of the best individual oscillates quite wildly for the duration of the 300 generations. With such a high crossover rate, it is evident that good solutions are being broken up at a higher rate and the recombination has resulted in an individual with poorer fitness value. Another point of interest can be found in the plot associated with P_c = 0.4. Note the spike in the plot very early on in the evolution process where a good solution with a fitness value greater than 900 was obtained. This solution was subsequently destroyed in the following generation; with the fitness of the best individual falling to a value close to 800. As the evolution progressed, an even better individual was produced. Subsequently, a further improved individual was obtained after 163 generations. In terms of the plot associated with P_c = 0.6, notice that there is a slight oscillation (very much smaller than those associated with P_c = 0.8) in the plot throughout the 300 generations.

At the end of our tuning process involving the three parameters

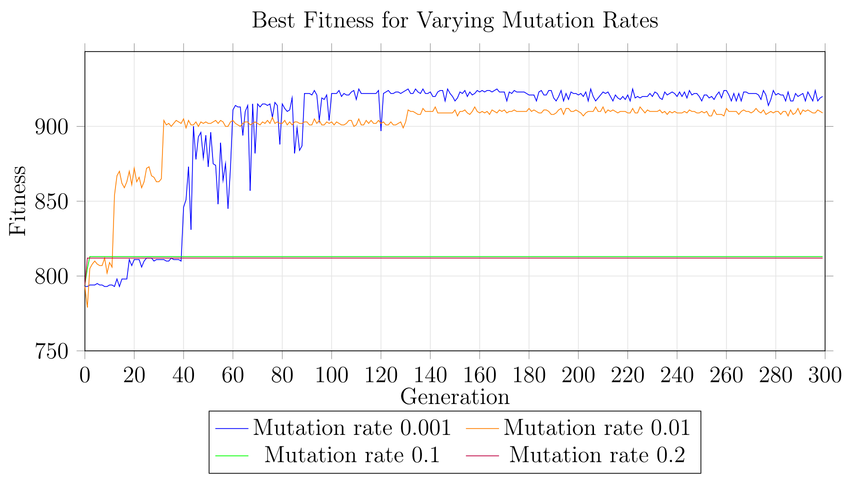

Fig. 12. Tuning Mutation Probability (Sim 1): plot for best fitness from each generation for each mutation probability.

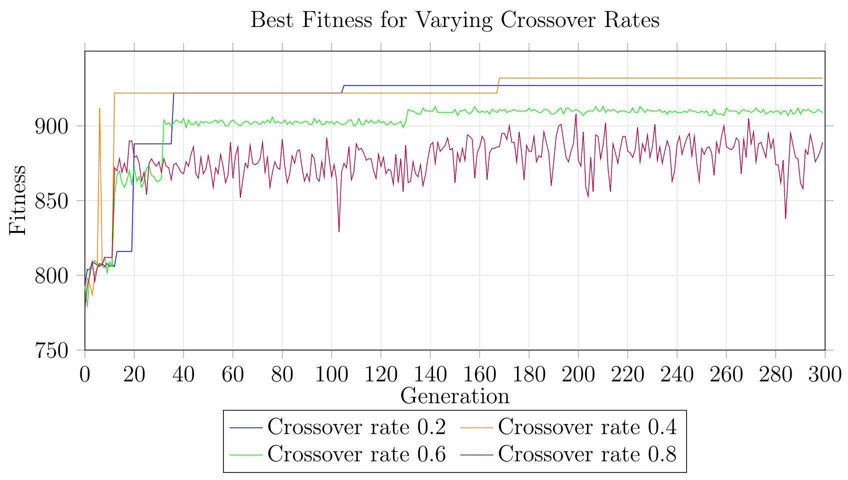

Fig. 13. Tuning Crossover probability (Sim 1): plot for best fitness from each generation for each crossover probability.

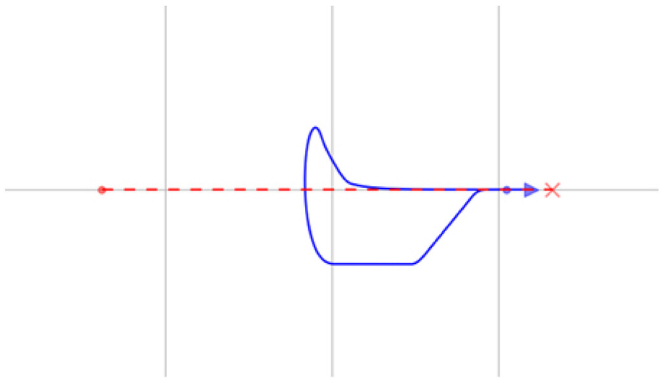

Fig. 14. Trajectory of the Blue Agent vs Red Agent from Sim 1 with the highest fitness upon termination of the GA using the tuned parameter values for population size(200), crossover (0.6) and mutation probability(0.001) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

above, the results are analysed to identify the set of tuned parameters associated with the task which involved the Blue Agent (DSTG-AT) attempting to complete a stern conversion against a Red Agent flying in a straight line. In our case study, these values include: population size = 200; mutation probability = 0.001 and crossover probability = 0.6. The optimal individual has a fitness value of 946. Fig. 14 shows the trajectory of the Blue Agent with the highest fitness upon termination of the GA using these parameter values for population size, crossover and mutation probability.

The change of states associated with the Blue agent in (14) is shown below:

- *Pure Pursuit* state from 0 to 10.2 s
- *Fly Relative Bearing* state from 10.3 s to 44.5 s
- *Flying Offset* state from 44.6 s to 94.6 s
- *Fly Relative Bearing* state from 94.7 to 94.7 s (immediate change out of the state)
- *Pure Pursuit* state from 94.8 s to 250 s (the end of the simulation)

For comparison, the resulting optimal trajectory for a blue agent evolved against a red agent attempting to perform a stern conversion, using the evolution parameters that were tuned against the straight-line agent is shown in Fig. 15. This solution corresponds to a fitness value of 955 and results in a successful stern conversion.

To provide a point of comparison for the evolved exemplars to a more traditional search technique, Iterated Local Search (ILS) was also

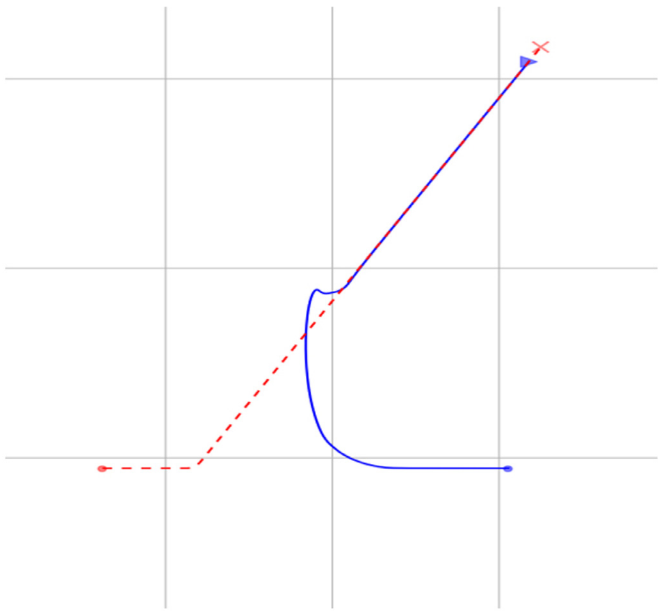

Fig. 15. Trajectory of the Blue Agent vs Red Agent from Sim 2 with the highest fitness upon termination of the GA using the tuned parameter values for population size(200), crossover (0.6) and mutation probability(0.001) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

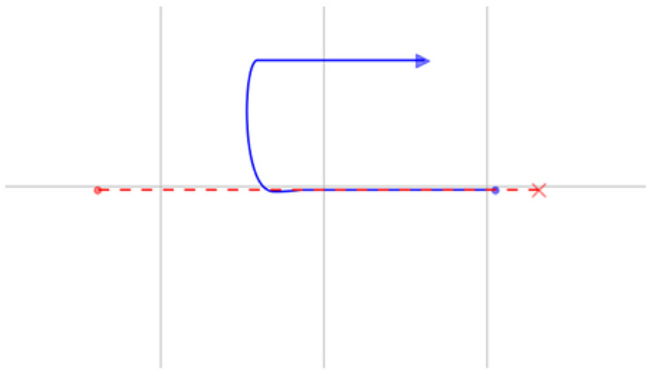

Fig. 16. Trajectory of the Blue Agent vs Red Agent from Sim 1 with the highest fitness upon termination of the ILS for 60,000 iterations (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

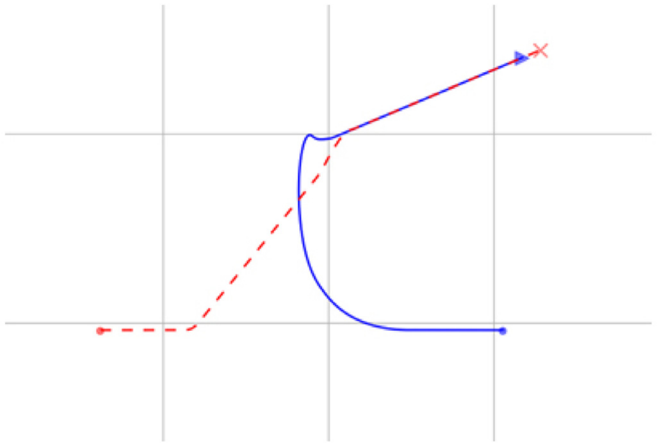

Fig. 17. Trajectory of the Blue Agent vs Red Agent from Sim 2 with the highest fitness upon termination of the ILS for 60,000 iterations (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

Table 7
Associated parameters values used in Figs. 14–17 .

Algorithm	Parameter	Value
GA	Population size	200
	Maximum generations	300
	Selection operator	Stochastic universal sampling
	Crossover operator	One point crossover
	Crossover probability (P_c)	0.6
	Mutation operator	Gaussian mutation
	Mutation probability (P_m)	0.001
	Elitism	Single best carried over
ILS	Local search iterations	300
	Algorithm iterations	200
	Local neighbour step strength	0.01
	Perturbation strength	0.1

Table 8
The transition parameters for the Stern Conversion FSM (DSTG-AT).

Problem	GA		ILS	
	Fitness	Evaluations	Fitness	Evaluations
Sim 1	946	60,000	818	60,000
	943	60,000	813	60,000
	940	60,000	810	60,000
	930	60,000	809	60,000
	926	60,000	808	60,000
Sim 2	955	60,000	948	60,000
	949	60,000	948	60,000
	948	60,000	948	60,000
	948	60,000	948	60,000
	948	60,000	948	60,000

applied to the Sim 1 and Sim 2 scenarios. To make for a fair comparison, the number of local search iterations and algorithm iterations were set so as to result in the same total number of simulation-based evaluations as a run of the evolutionary process (60,000). For comparison and reproducibility, the parameters for both the evolutionary approach and ILS are provided in (7). Trajectories corresponding to the best solution found by ILS for Sim 1 and Sim 2 are shown in (16) and (17) respectively. For Sim 1 (16) it can be seen that the solution found by ILS is sub-optimal, the blue aircraft ending up behind the red, as intented, but flying at an offset distance. For Sim 2 (17) a stern conversion manouvre was found by ILS. Due to the stochastic nature of both evolution and ILS, five runs of each, on both Sim 1 and Sim 2 were undertaken. The numeric results are presented in (8), indicating the fitness achieved by the optimal solution in each run, along with the number of evaluations (60,000 in each case). To given an indication of concrete computing resources required for these runs, on an Intel i7 - based CPU with 16MB of RAM one run of Sim 1 optimisation is in the range of 350 s, and for Sim 2 in the range of 400 s. This is irrespective of whether evolution or ILS is used, as the bulk of the processing resources are used in the simulation-based evaluation of the individual solutions.

Other GA design considerations to be addressed as part of the tuning process are strategies for parent selection and replacement. As shown in Table 5, there are two categories of parent selection strategies to ensure survival of the fittest individuals, namely fitness proportionate and rank-based selection. A point to note here is that fitness proportionate selection methods are susceptible to both premature convergence and stalled evolution [42]. Rank-based selection methods are designed to address these shortcomings. Probability associated with parent selection and replacement strategy impact on the trade-off between exploration and exploitation. Exploration is enhanced with using smaller values (for both) owing to a slower convergence while bigger values will encourage exploitation. Typically, a smaller value for both is preferred, allowing for more exploration and hopefully avoiding premature convergence leading to a suboptimal solution, even though the time to convergence will take much longer. However, encouraging exploration increases the probability of losing good solutions in the current pool of solutions and a mechanism to counteract this situation is to incorporate elitism – a method for that prevents random destruction (by crossover or mutation) and ensures that the best individuals are preserved in the population from one generation to the next and their building blocks being passed on to subsequent generations. The number of elite individuals should not be too big, otherwise the population will likely deteriorate.

Lastly, the termination criterion is another consideration for the GA process. Typical criteria for terminating the cycle of evolution include:

- The GA has reached a pre-defined maximum number of generations
- Maximum allocated resources (computation time) have been used
- The population has converged to an optimum, as determined using:
- Change in fitness value between two consecutive generations is less than δ which is a very small number (e.g. 0.001)
- Approach proposed by Srinivas and Patnaik [43]. This involved checking that the difference between the average fitness (*favg*) and maximum fitness value (*fmax*) of two consecutive generations is smaller than a very small number, δ.
- Combinations of the above

4.9. Step 4 evaluation considerations: repeatability and reproducibility of GA-based approach

As mentioned earlier, GA is stochastic algorithm and thus care must be taken in algorithm design and evaluation to ensure repeatability and reproducibility. The repeatability of a GA-based algorithm is defined as the consistency of the algorithm to produce very similar results under a set of the same experimental conditions (i.e. same input and using the same set of parameter settings, fitness evaluation and representation) from repeated runs. Unlike a deterministic algorithm that will produce the same exact result given the same input, there will be some small variations from each execution of a GA-based algorithm. The first point in a GA-based approach with randomness is associated with the generation of the initial population – typically done randomly for every gene associated with each individual in the population. For repeatability, we need the same initial population for each repeated execution of the algorithm. Typically the approach taken here is to pass the same seed to the random number generator in the module for initialisation of the initial population. The seed ensures the same set of random numbers are generated – resulting in exact replicas of individuals in the initial population.

Reproducibility is associated with the ability of the algorithm to get the similar results under a set of different experimental conditions (e.g.

– a different initial population). To evaluate the GA-based approach here, the approach must be evaluated over several different sets of random numbers to guarantee the robustness of the approach. By using a different seed, we can ensure that each execution run of the algorithm uses a set of different individuals. Reproducibility can then by examined in a quantitative manner – in terms of the statistical dispersion of the results.

5. Discussion

In this study, the agent-based behaviour model is captured using FSMs, consisting of states and transitions. The GA-based approach is to evolve a sequence of transitions between states in DSTG-AT such that a stern conversion manoeuvre is successfully executed by the Blue Agent (DSTG-AT) against a Red Agent. Note that the chromosome we designed represents the set of all possible transitions in DSTG-AT where transition conditions are captured in terms of kinematic properties of the two aircraft (Blue and Red), namely, the position (**p**), velocity (**v**) and acceleration (**a**). Only low level simplistic information has been captured, yet through the evolution process, the final solution captured the sequence of transitions for successfully completing a stern conversion, hence demonstrating the potential of such approaches for exploring automatic generation of behaviour models of CGFs where constructing any realistic model would be a huge task. These sort of findings have also been shown in work of other researchers, hence demonstrating the potential of GA in terms of generating emerging or unexpected behaviours. Ranjeet, Hingston, Lam & Masek [44] showed that more complex strategies can emerge from a GA-based approach with chromosome representations comprising of simplistic information in their study associated with analysis of key installation protection involving MANA - cellular automaton combat simulator from the New Zealand Defence technology agency. Using a GA-based approach, Hamblin & Hurd [45] applied GA to the *Sir Philip Sidney game*, a simple model of biological signalling and found an unexpected solution with higher pay-offs at equilibrium; namely a solution whereby individuals would never send out a signal at all.

Even though in practice, GA can perform robustly under a range of values for its parameters, it is important to tune the parameter values to some extent; bearing in mind that there is a trade-off between the effort needed to tune versus the quality of solutions and the performance of the GA-based algorithm. Owing to the unknown non-linear dependencies which commonly exist between these parameters, the effect of adjusting parameters associated with the algorithm is not fully understood. Given the difficulties in determining these parameter values, this task is of much research interest; with work investigating the use of self-adaptive parameters to address this problem. Finally, a point to bear in mind is that we must be careful in terms of generalizing the performance of parameters across problem instances, problem domains, and the length of the chromosome of the representation chosen. For example, considering the probability of changing a gene, P_m, the more genes in a chromosome, the less likely that it will make it into the next generation without mutation, up to a point where good solutions may be lost.

Potential limitations with GA-based approaches in this problem domain include premature convergence with the final solution being a local optimal and that there may be a better solution out there but your evolutionary process has not found it yet (i.e. the current best is the best-so-far). Given that most complex problems are multi-modal, the question is how important is being able to find the global optimal. Is there value in a sub-optimal solution (local optimal) which may still be useful from the perspective of exploring and identifying novel combat tactics for the blue agent? On the other hand, given the right set of parameters, representation, fitness evaluation for the problem and sufficient run time, theoretically it is possible to obtain the global optimal.

6. Conclusion

The problem associated with evolving agent-based behaviour models associated with novel combat tactics in military operations is a multi-criteria and a multi-modal problem. It also involves a co-adaptive environment with individuals making tactical moves and their opponents making counter moves, forming one part of a dynamic environment. With such problems, no one knows exactly what the optimal solution looks like, there is very little heuristic information to guide researchers to analytically and systematically develop solutions and, as the search space is infinitely large, brute forced searches are not viable options. The application of a GA-based approach is particularly suitable in the first instance to address the task of automatically evolving novel combat tactics for CGFs as this task is complex, mathematically intractable and bears close similarities with generating models associated with evolutionary game theory. Investigations involving Multi-objective and Co-evolutionary algorithms are natural subsequent research directions to be undertaken to study this domain.

In summary, we have discussed our GA-based approach and demonstrated its potential to evolve behaviour models for CGFs, in comparison to more traditional approaches such as ILS. We also have attempted to shed light on some of the methodological difficulties associated with using GA to evolve combat tactical models of CGFs as well as providing workable starting points for researchers intending to use GA in their own attempts in this problem domain. GA can be a powerful approach if used properly, and is one of the best ways to find possible solutions to problems where no one knows exactly what the optimal solution looks like and with very little heuristic information to guide researchers to analytically and systematically develop solutions.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This research is supported by the Defence Science and Technology Group, Australia under the Modelling Complex Warfighting Strategic Research Investment (Grant no. 7840/G1003574). The authors would also like to acknowledge the support of Mr. Martin Cross (DST Group) who sponsored this research.

References

[1] Mittal S, Doyle MJ, Watz E. Detecting intelligent agent behavior with environment abstraction in complex air combat systems. Systems conference (SysCon), 2013 IEEE international. IEEE; 2013. p. 662–70.
[2] Roessingh J., Rijken R., Merk R., Meiland R., Huibers P., Lue T., et al. Modelling CGFs for tactical air-to-air combat training motivation-based behaviour and machine learning in a common architecture2011;.
[3] Toubman A, Roessingh JJ, van Oijen J, Løvlid RA, Hou M, Meyer C, et al. Modeling behavior of computer generated forces with machine learning techniques, the nato task group approach. Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on. IEEE; 2016. p. 001906–11.
[4] Juan AA, Faulin J, Grasman SE, Rabe M, Figueira G. A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems. Oper Res Perspect 2015;2:62–72.
[5] Juan A, Chica M, de Armas J, Kelton W. Simheuristics: a method of first resort for solving real-life combinatorial optimization problems. Keynote papers of the OR58 annual conference, Portsmouth, UK. 2016. p. 147–56.
[6] Jang SH, Kim TY, Kim JK, Lee JS. The study of genetic algorithm-based task scheduling for cloud computing. Int J Control Autom 2012;5(4):157–62.
[7] Larranaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S. Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 1999;13(2):129–70.
[8] Lin C-C, Liu Y-T. Genetic algorithms for portfolio selection problems with minimum transaction lots. Eur J Oper Res 2008;185(1):393–404.
[9] Kazarlis S, Petridis V, Fragkou P. Solving university timetabling problems using advanced genetic algorithms. GAs 2005;2(7):8–12.

[10] Yao J, Huang Q, Wang W. Adaptive human behavior modeling for air combat simulation. Distributed simulation and real time applications (DS-RT), 2015 IEEE/ACM 19th international symposium on. IEEE; 2015. p. 100–3.
[11] Masek M, Lam CP, Benke L, Kelly L, Papasimeon M. Discovering emergent agent behaviour with evolutionary finite state machines. International conference on principles and practice of multi-agent systems. Springer; 2018. p. 19–34.
[12] Mitchell M, Forrest S, Holland JH. The royal road for genetic algorithms: fitness landscapes and ga performance. Proceedings of the first European conference on artificial life. 1992. p. 245–54.
[13] Shaabani H, Kamalabadi IN. An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem. Comput Ind Eng 2016;99:189–201.
[14] Panadero J, Juan AA, Mozos JM, Corlu CG, Onggo BS. Agent-based simheuristics: extending simulation-optimization algorithms via distributed and parallel computing. Proceedings of the 2018 winter simulation conference. IEEE Press; 2018. p. 869–80.
[15] Fu MC. Optimization for simulation: theory vs. practice. INFORMS J Comput 2002;14(3):192–215.
[16] Amaran S, Sahinidis NV, Sharda B, Bury SJ. Simulation optimization: a review of algorithms and applications. Ann Oper Res 2016;240(1):351–80.
[17] Figueira G, Almada-Lobo B. Hybrid simulation–optimization methods: a taxonomy and discussion. Simul Model Pract Theory 2014;46:118–34.
[18] Jackson I, Tolujevs J, Reggelin T. The combination of discrete-event simulation and genetic algorithm for solving the stochastic multi-product inventory optimization problem. Transp Telecommun J 2018;19(3):233–43.
[19] Noroozi A, Mokhtari H. Scheduling of printed circuit board (PCB) assembly systems with heterogeneous processors using simulation-based intelligent optimization methods. Neural Comput Appl 2015;26(4):857–73.
[20] Grefenstette JJ. Optimization of control parameters for genetic algorithms. IEEE TransSystManCybern 1986;16(1):122–8.
[21] De Jong K. Parameter setting in EAs: a 30 year perspective. Parameter setting in evolutionary algorithms. Springer; 2007. p. 1–18.
[22] Shaw RL. Fighter combat. Naval Institute Press; 1985.
[23] Holland JH. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press; 1992.
[24] Fleming PJ, Purshouse RC. Evolutionary algorithms in control systems engineering: a survey. Control EngPract 2002;10(11):1223–41.
[25] Koza J.R.. Genetic programming: vol. 1, on the programming of computers by means of natural selection, vol. 1. 1992.
[26] Goldberg D, Deb K, Korb B. Messy genetic algorithms: motivation, analysis, and first results. Complex Syst 1989(3):493–530.
[27] Whitley D, Mathias K, Fitzhorn P. Delta coding: an iterative search strategy for genetic algorithms. ICGA. 91. 1991. p. 77–84.
[28] Maaranen H, Miettinen K, Penttinen A. On initial populations of a genetic algorithm for continuous optimization problems. J Glob Optim 2006;37(3):405. https://doi.org/10.1007/s10898-006-9056-6.
[29] Magalhaes-Mendes J. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. WSEAS TransComput 2013;12(4):164–73.
[30] Abdoun O., Abouchabaka J., Tajani C.. Analyzing the performance of mutation operators to solve the travelling salesman problem. arXiv:12033099 2012;.
[31] Koenig AC. A study of mutation methods for evolutionary algorithms. University of Missouri-Rolla; 2002.
[32] Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms. 1. Elsevier; 1991. p. 69–93.
[33] Ramirez M, Papasimeon M, Lipovetzky N, Benke L, Miller T, Pearce AR, et al. Integrated hybrid planning and programmed control for real time UAV maneuvering. Proceedings of the 17th international conference on autonomous agents and multiAgent systems. International Foundation for Autonomous Agents and Multiagent Systems; 2018. p. 1318–26.
[34] De Jong KA, Spears WM. An analysis of the interacting roles of population size and crossover in genetic algorithms. International conference on parallel problem solving from nature. Springer; 1990. p. 38–47.
[35] Goldberg D. Genetic algorithms in search, optimization and machine earning. Addison-Wesley; 1989.
[36] Alander JT. On optimal population size of genetic algorithms. 1992 Proceedings computer systems and software engineering. IEEE; 1992. p. 65–70.
[37] Goldberg DE, Deb K, Clark JH. Genetic algorithms, noise, and the sizing of populations. Urbana 1991;51:61801.
[38] Harik GR, Lobo FG. A parameter-less genetic algorithm. Proceedings of the 1st annual conference on genetic and evolutionary computation-Volume 1. Morgan Kaufmann Publishers Inc.; 1999. p. 258–65.
[39] Pelikan M, Goldberg DE, Cantú-Paz E. Bayesian optimization algorithm, population sizing, and time to convergence. Proceedings of the 2nd annual conference on genetic and evolutionary computation. Morgan Kaufmann Publishers Inc.; 2000. p. 275–82.
[40] Kale K. Advances in computer vision and information technology. IK International Pvt Ltd.; 2008.
[41] Carroll DL. Chemical laser modeling with genetic algorithms. AIAA J 1996;34(2):338–46.
[42] Hopgood AA. Intelligent systems for engineers and scientists. CRC Press; 2011.
[43] Srinivas M, Patnaik LM. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans Syst Man Cybern 1994;24(4):656–67.
[44] Ranjeet TR, Hingston P, Lam C-P, Masek M. Analysis of key installation protection using computerized red teaming. Proceedings of the thirty-fourth Australasian computer science conference-Volume 113. Australian Computer Society, Inc.; 2011. p. 137–44.
[45] Hamblin S, Hurd PL. When will evolution lead to deceptive signaling in the sir philip sidney game? TheorPopulBiol 2009;75(2–3):176–82.