Comparative Analysis of Metaheuristic Algorithms for Procedural Race Track Generation in Games

Sana Alyaseri
https://orcid.org/0009-0004-8650-0109
Whitecliffe College, New Zealand

Andy Conner
https://orcid.org/0000-0002-4046-5959
Auckland University of Technology, New Zealand

ABSTRACT

Procedural Content Generation (PCG) aims to automatically generate the content of games using algorithmic approaches, as this can reduce the cost of game design and development. PCG algorithms can be applied to all elements of a game, including terrain, maps, stories, dialogues, quests, and characters. A wide variety of search algorithms can be applied to PCG problems; however, those most often used are variations of evolutionary algorithms. This study focuses on comparing three metaheuristic approaches applied to racetrack games, with the specific goal of evaluating the effectiveness of different algorithms in producing game content. To that end, a Genetic Algorithm (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) are applied to a game-level design task to attempt to identify any discernible differences in their performance and identify whether alternative algorithms offer desirable performance characteristics. The results of the study indicate that both the ABC and PSO approaches offer potential advantages to Genetic Algorithm implementation.

KEYWORDS

Procedural Content Generation (PCG), Genetic Algorithms (GAs), Genetic algorithms, Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Metaheuristics

INTRODUCTION

Games are often considered a suitable test environment for artificial intelligence techniques, and game developers often focus on finding approaches that can automatically create computer game content to help minimize the load on developers (Zhang et al., 2022). Procedural content generation (PCG) can be used to build all types of game content, including levels, environments, and components. As such, it can be considered a tool that aids the designer in creating this content by offering speed advantages in comparison to manually designing the content and potentially through the discovery of novel content. PCG can create a realistic and aesthetically pleasing user experience (Korn et al., 2017) and has been effectively used in a wide range of commercial games. Such games have their content algorithmically developed either during the design process or, in some cases, during runtime, rather than being designed by humans. A wide range of approaches have been used (Barriga, 2019;

DOI: 10.4018/IJAMC.350330

Zhang et al., 2022), and there is a growing interest in search-based approaches (Blasco et al., 2023; Skjærseth et al., 2021; Zafar et al., 2020; H. Zhang et al., 2018).

The literature shows that evolutionary approaches, such as genetic algorithms (GAs), are one of the most widely used metaheuristic methods in search-based PCG (SBPCG); they have been shown to be successful in solving a variety of complicated problems (Alyaseri et al., 2024). In other domains of study, comparisons of metaheuristic algorithm performance have shown that in many cases, the popularity of GAs is not entirely justified. Although GAs excel in various tasks, in some contexts they might exhibit limitations, such as slow convergence or becoming trapped in local optima. This has motivated the exploration of alternative algorithms (Connor & Tilley, 2016; Panda, 2018; Schmidt et al., 2018; Youssef et al., 2001).

Therefore, the main purpose of this study was to compare GAs with two different metaheuristic algorithms—artificial bee colony (ABC) and particle swarm optimization (PSO)—in the context of PCG for computer games. These algorithms have demonstrated promising characteristics in other domains, such as faster convergence for PSO and improved exploration of ABC. PSO in particular offers advantages because of its simplicity and ease of implementation. PSO also leverages a form of collective memory by considering both the best positions experienced by individual particles and those discovered by the entire swarm. This information sharing helps particles explore the search space more effectively and avoid getting stuck in the local optima. In contrast, the GA primarily focuses on individual selection and crossover, potentially discarding valuable information from less successful individuals (Couceiro et al., 2016). ABC also possesses a distinct exploration strategy. In the ABC algorithm, both onlookers and employed bees play a role in exploring the search space. Employed bees exploit the food sources (potential solutions) they discovered in previous iterations. Onlookers, based on the employed bees' dancing behavior (a form of information sharing), select promising food sources for further exploration.

Furthermore, scout bees handle diversification by randomly generating new food sources, ensuring the algorithm does not get stuck in local optima. This combination of exploitation (employed bees), informed exploration (onlookers), and random exploration (scouts) contributes to ABC's potential effectiveness in finding good solutions (Abu-Mouti & El-Hawary, 2012). The present study specifically provides an example of automatically generating race tracks for a racing game. The aim was to identify whether alternative algorithms, such as ABC and PSO, with their potential advantages observed in other domains, can offer similar benefits when applied to the creation of game content, such as race tracks.

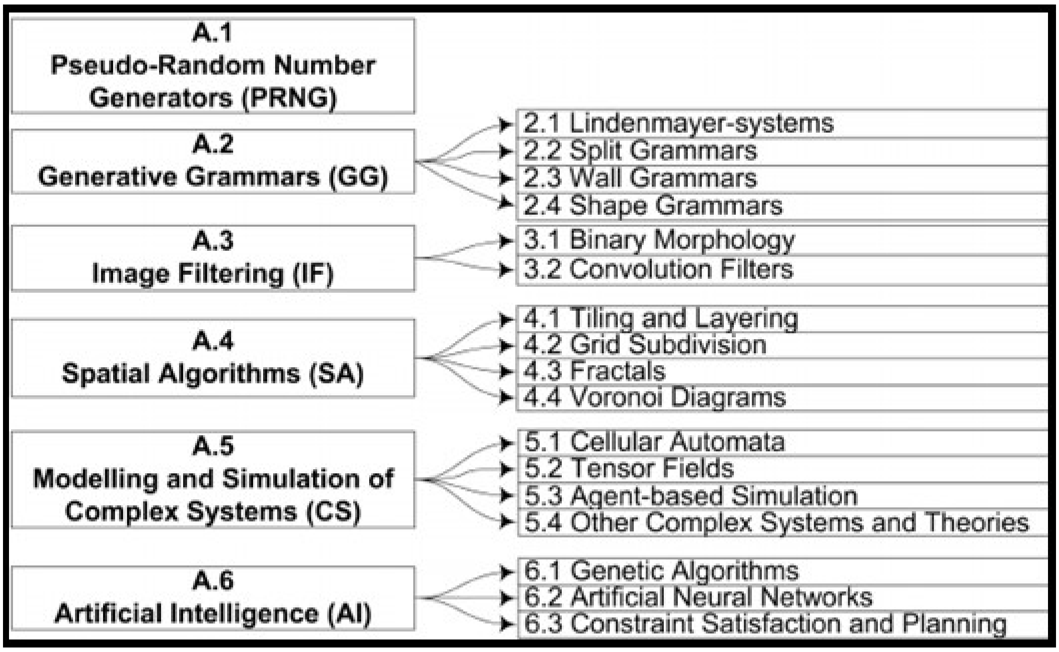
The remainder of this article is organized as follows. We first provide an overview of the literature supporting this study. Next, we explain our methodological approach, which includes the PCG task and implementation of the selected metaheuristics. We then provide details about the performance comparisons of various search algorithms in generating race tracks for a racing game. We conclude with a discussion of the results and their implications along with suggestions for future research.

BACKGROUND AND RELATED WORK

PCG is the automated creation of game content through algorithmic processes (Yannakakis et al., 2018). Hendrikx et al. (2013) proposed a comprehensive taxonomy that categorizes common PCG methods into six sections, as illustrated in Figure 1. Although various projects in both the gaming industry and research have explored different methods for generating content across multiple levels of abstraction, in this study we specifically focus on evaluating the justification for the popularity of evolutionary approaches within the context of search-based methods.

The chosen algorithms for this study fall under the category of artificial intelligence in Hendrikx et al.'s (2013) taxonomy because the primary goal is to compare similar approaches. Although a detailed examination of all the methods is beyond the scope of this research, the taxonomy itself serves as a valuable frame of reference for understanding the landscape of PCG methods. In addition

Figure 1. Procedural content generation methods taxonomy



to Hendrikx et al.'s taxonomy, another significant taxonomy, developed by Zhang (2022), classifies PCG methods into three main groups: search-based, traditional, and machine learning approaches.

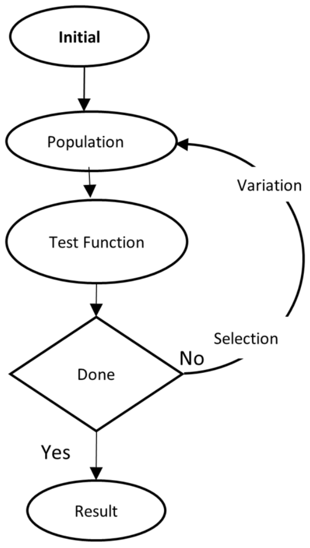
It is notable that metaheuristic approaches were not explicitly addressed in Zhang (2022) study, potentially because they categorized such approaches as a subset of search-based methods. This omission may stem from the observed similarities between the techniques used in the search-based methods and metaheuristics. Despite this gap, in this study our aim was to focus on a detailed comparison of evolutionary approaches in a broader context of SBPCG methods.

The algorithms selected for this study are classified as metaheuristic search algorithms. Togelius et al. (2011) described the concept of search-based SBPCG as an advanced case of the more generalized "generate and test" approach. Although the literature normally identifies this approach as a method that works on a population, as shown in Figure 2, this reflects the popularity of evolutionary algorithms, including GAs, although it can also include approaches that are not population based.

Approaches like the one depicted in Figure 2 have been extensively used in literature. For example, Ferreira et al. (2014) applied a multi-population GA to Super Mario Bros. to grow four game features separately: the ground, block, opponents, and coins. In this example, each element has its own population and fitness function, and the approach joins the best individual in the population from each element to build the complete level. Many other examples exist with applications in relation to platform games (Moghadam & Rafsanjani, 2017), puzzle games (Pereira & Toledo, 2017), nonplayer characters (Kim et al., 2017), game narratives (de Lima et al., 2022), and crafted bosses in the Kromaia video game (Blasco et al., 2023). All of these studies reached similar conclusions: that procedurally generated content can meet the design intent of the game and contribute to the creation of playable levels.

Given the apparent success of evolutionary approaches, however, there is no reason why game content cannot be generated by different algorithms. In many other domains, research has shown that other algorithms can offer improved performance, although some authors have indicated that there is a lack of standardization in the evaluation of algorithms (e.g., Dokeroglu et al, 2019), thereby

Figure 2. Search-based procedural content generation



emphasizing the importance of comparative studies in a given context. There is little evidence in the literature for the use of other metaheuristic approaches applied to the creation of content for games (Togelius et al., 2010).

De Carli et al. (2011) conducted a study of both classical and current techniques that yielded numerous algorithms that can be applied to game content creation. Although the scope of their survey was broad, it did not consider that metaheuristic approaches could be used at all. This suggests that the focus on metaheuristic approaches is relatively recent, but it does not explain the focus on evolutionary algorithms. The authors of another review, which included metaheuristic techniques, went so far as states that SBPCG can use algorithms such as simulating annealing and PSO (Togelius et al., 2011). However, the authors also did not provide any actual examples of their applications in PCG, and a direct search of the literature yielded only limited evidence that alternative metaheuristics have been used in SBPCG. For example, simulated annealing, Markov chains (Kim & Crawfis, 2015), and PSO have been used to automatically generate two-dimensional graphical characters (Fister et al., 2015) as well as game levels for an infinite-platform game (de Pontes et al., 2022). Other algorithms have also been applied to gaming contexts. For example, Tabu Search has been applied to map generation for tabletop games (Grichshenko et al., 2020) and race tracks (Prasetya & Maulidevi, 2016). However, only a limited amount of research has explored the use of metaheuristic methods in computer games aside from evolutionary algorithms (Alyaseri et al., 2024).

In the PCG literature, irrespective of whether a study is based on evolutionary algorithms or an alternative, few studies have compared the outcomes with those provided by any other metaheuristic algorithm; neither is there strong justification for selecting these approaches on the basis of an empirical understanding of their performance characteristics. Some researchers have conducted comparative studies of the performances of metaheuristic algorithms. For example, one study conducted to generate race tracks in video games used Tabu Search and GA using two different approaches. The results mostly showed the variations between the two approaches in terms of the speed convergence, whereby the GA was faster than the Tabu Search (Prasetya & Maulidevi, 2016). However, this project is just one instance of a comparison with limited scope. It appears that there is no considerable emphasis on evaluating alternative algorithms that can be used in SBPCG.

In contrast to PCG, the use of metaheuristic algorithms in other fields has resulted in considerably more diversified representations of algorithms. For example, Dokeroglu et al. (2019) listed 14 algorithms that have been used to generate new solutions to a range of problems; however, this list is far from exhaustive. Although GAs are still common in other domains, the use of other algorithms is much more prevalent, and a range of comparative studies have been undertaken in different areas over a long time (Collette et al., 2000; Connor & Shea, 2000; Leite da Silva et al., 2011; Gupta et al., 2021; Kannan et al., 2005; Reche-López et al., 2009). As a result, a broader understanding of the range of algorithms that solve a significant number of well-defined issues, recognize their strengths and limitations, and determine their appropriate settings will be used to support solution optimization. To date, only a limited number of studies have compared different algorithms in the context of PCG in games. Metaheuristic algorithms belong to a domain of algorithms inspired by natural phenomena. These are known as *nature-inspired algorithms*. In this study, we did not depend on a specific classification when we selected the algorithms for comparison, because some algorithms may belong to more than one category.

Categorization depends on focus, emphasis, and perspective (Dhal et al., 2020; Fister et al., 2013). Researchers have developed and evaluated a range of algorithms in this field, each of which has a distinct benefit in dealing with specific types of problems while also having disadvantages. However, the algorithms included as subjects in this study were GAs, PSO, and ABCs (Dalwani & Agarwal, 2018). The latter two are examples of swarm algorithms, in contrast to the evolutionary foundation of GAs. GAs are the first approach used in this study and are included as a baseline for comparison against which other approaches are evaluated. It is the most commonly used evolutionary algorithm and was inspired by the concepts of natural selection and the survival of the fittest. The algorithm repeatedly modifies the population of the candidate solutions. For each generation, individuals are selected from the current population as parents, which are used to produce children for the next generation. The more fit candidate solutions in a population are more likely to be selected as parents, which is how the principle of survival of the fittest is embodied, and children are produced through modeling the genetic operators of crossover and mutation (Kramer & Kramer, 2017). Over successive generations, the population evolves toward an optimal solution.

GAs have been applied to a wide range of problems in operations research, engineering, and science (Connor, 1996; Ghaheri et al., 2015; Man et al., 1996; Manning et al., 2013; Paszkowicz, 2013; Potvin, 1996; Wang et al., 2003). Video games broadly use GAs in search-based PCG, which are particularly useful for generating content in massive multiplayer online games (Alves et al., 2018), levels of platform games (Mourato et al., 2011), tower defense games (Kraner et al., 2021), quest generation (de Lima et al., 2022), and the design of levels in a dungeon crawler game (Liapis et al., 2013).

ABC is the second metaheuristic approach we used in this study. It is inspired by animal behavior and falls into the category of swarm intelligence (SI). SI is defined as a group of techniques that uses a population of workers, particles, or agents working together to find an optimal (or near-optimal) solution to the problem at hand. In the ABC algorithm, the behavior of this population consists of simulated honeybees searching for food sources (Karaboga et al., 2014). Self-organization and the division of labor are two fundamental bases that are adequate for producing intelligent swarm behavior (Alyaseri & Aljanaby, 2014). These fundamentals are powerfully and clearly seen in honeybee colonies, along with satisfaction principles.

Although the ABC algorithm is not as popular as GAs, it is still widely applied, and many researchers have found it to be an effective technique for solving various optimization problems. For example, it improves edge detection in digital image processing (Liu & Tang, 2017). This algorithm was specifically developed in the search method for neighboring edge points to obtain the final edge detection figure. ABC was also used to solve the antenna array design issue and was compared with four other algorithms; the results and analysis of this method show that it is appropriate for solving antenna-design problems (Wang et al., 2019). In practical engineering optimization problems, ABC has also been used as an efficient approach to solve interval optimization problems. L. Zhang et al. (2018) introduced a novel approach that uses ABC to enhance the generation of perfect interval credibility in these scenarios.

In relation to PCG, the ABC algorithm has rarely been applied to video games. Mora et al. (2021) referred to the ABC algorithm in their study, but they ultimately applied a different algorithm, namely, the artificial flora algorithm. In a select few studies that used comparative analysis, Yücel and Suürer, (2020) compared the ABC algorithm with both PSO and the firefly algorithm. Their research was conducted in the context of an abstract game scenario wherein players manipulate daisies within a grid to capture swarms in a jar. Although their results showed that the firefly algorithm outperformed both the PSO and ABC algorithms, there were cases in which ABC outperformed PSO, and vice versa. However, in this study we used an algorithm for game mechanics rather than to generate specific game content, so it falls outside the area of PCG.

PSO also has demonstrated good performance in many applications, such as the ABC algorithm, which falls into the SI category. PSO mimics the social behavior of flocking birds and schooling fish. The search for the optima is performed by updating the fitness value and position of the particles in each cycle. The position is usually updated by imparting velocity to it. Velocity is a vector of the sum of the partial current position, previously the best position, and the global best position accomplished by any particle in the search space. This iterative process continues until the best global position is reached (Pal et al., 2011).

PSO has been applied to a wide range of problems, including robot path planning (Sahu et al., 2018), ship design (Zheng et al., 2021), and a limited number of applications related to video games (de Araújo et al., 2020). There have been a limited number of applications of PSO as an algorithm in PCG tasks. For example, de Pontes et al. (2022) not only showed that PSO can successfully be applied to the content generation of an infinite platformer game but also indicated that the algorithm offers advantages over GAs in terms of both effectiveness and efficiency.

GA, ABC, and PSO algorithms have been shown to be effective in optimizing solutions to a range of problems. Several studies have compared these algorithms, with varying results. For example, Kulkarni and Desai (2016) suggested that the ABC algorithm delivers a more accurate optimization than PSO; however, it takes a longer time to converge. In another study, Kanović et al. (2014) suggested that there was minimal performance variance among GA, PSO, and ABC. However, ABC occasionally experiences delayed convergence, which contrasts with the findings of Karaboga and Akay (2009), who demonstrated that the ABC algorithm surpasses other metaheuristics across various test functions. This discrepancy may be attributed to the problem Kanović et al. studied

Settles et al. (2003) investigated GAs and PSO in evolving neural networks and observed that GAs perform better on large networks, whereas PSO performs better on smaller networks. Indeed, there is little consensus yet as to whether any algorithm is consistently better, and any outcome is highly dependent on the application domain and implementation.

Nevertheless, studies of SBPCG have been limited to some algorithms, focusing on evolutionary computation, and have imported its terminology. However, only a few studies have experimented with heuristic search methods, and only a few studies have attempted this type of examination using several algorithms to identify how they may help fulfill PCG objectives. Research in other fields has shown that various algorithms have advantages over GAs, Examples are (Sahu et al., 2018), (Zheng et al., 2021), and (Liu & Tang, 2017). The present study is timely and relevant because we conduct an empirical evaluation of the different algorithms used in PCG applications.

METHOD

The purpose of this comparative study was to understand how well various metaheuristic search algorithms perform when used in PCG tasks to develop game levels. As we highlighted in the literature review, there is a strong emphasis on adopting GAs as an evolutionary method for PCG, even if alternative algorithms offer potential advantages in other applications. Given the nature of this study, the main component of the evaluation was quantitative, involving a comparison of the algorithms' effectiveness and efficiency, that is, their ability to find decent solutions based on a designed fitness function and their convergence speed to a solution.

Research Design

In this study, we used a computational approach to address our research objective. The primary methodology included the application of GA, PSO, and ABC to a race track game. The process involved implementing each algorithm, collecting relevant data, and systematically comparing their performance. Given the stochastic nature of the algorithms selected for this study, the evaluation process involved creating a race track. The study had an experimental design with a primary focus on determining significant differences among the algorithms. The experimental setup ensured that all the variables were held constant, with the only dependent variable being the different algorithms. This approach facilitated the attribution of identifiable differences to the specific algorithm under examination. To enhance the consistency of the comparison, each algorithm was operated with an equal population size of 600 individual solutions and experienced a fixed number of iterations.

Our initial experiments used conventionalized populations; however, these experiments did not yield significant performance improvement. Therefore, we investigated the effect of a larger population of 600 individuals over 200 iterations. This choice, although seemingly unconventional, was motivated by the inherent complexity of our problem. The problem domain involves discovering diverse race tracks that feature a variety of curves to enhance player engagement. We hypothesized that a larger population would provide broader exploration space, potentially leading to improved performance.

It is reasonable to posit that larger populations could offer advantages for all three algorithms under study: GAs, PSO, and ABC. For instance, GAs require a balance between maintaining diversity within the population and exploring new regions in the search space. A larger population size might facilitate this balance, in particular for intricate problems such as those we are addressing (Farahbakhsh & Kheirkhah, 2023). However, it is important to acknowledge the trade-off between the population size and computational cost. Larger populations inevitably require more computational resources.

In contrast, PSO is recognized for its ability to rapidly converge on solutions; therefore, smaller population sizes may be sufficient. Although maintaining a certain level of diversity is crucial for effective exploration, an excessively large population may not be necessary (Couceiro et al., 2016). Like PSO, ABC typically uses smaller populations and exhibits a rapid convergence, and thus a very large population is unlikely to offer significant benefits (Abu-Mouti & El-Hawary, 2012).

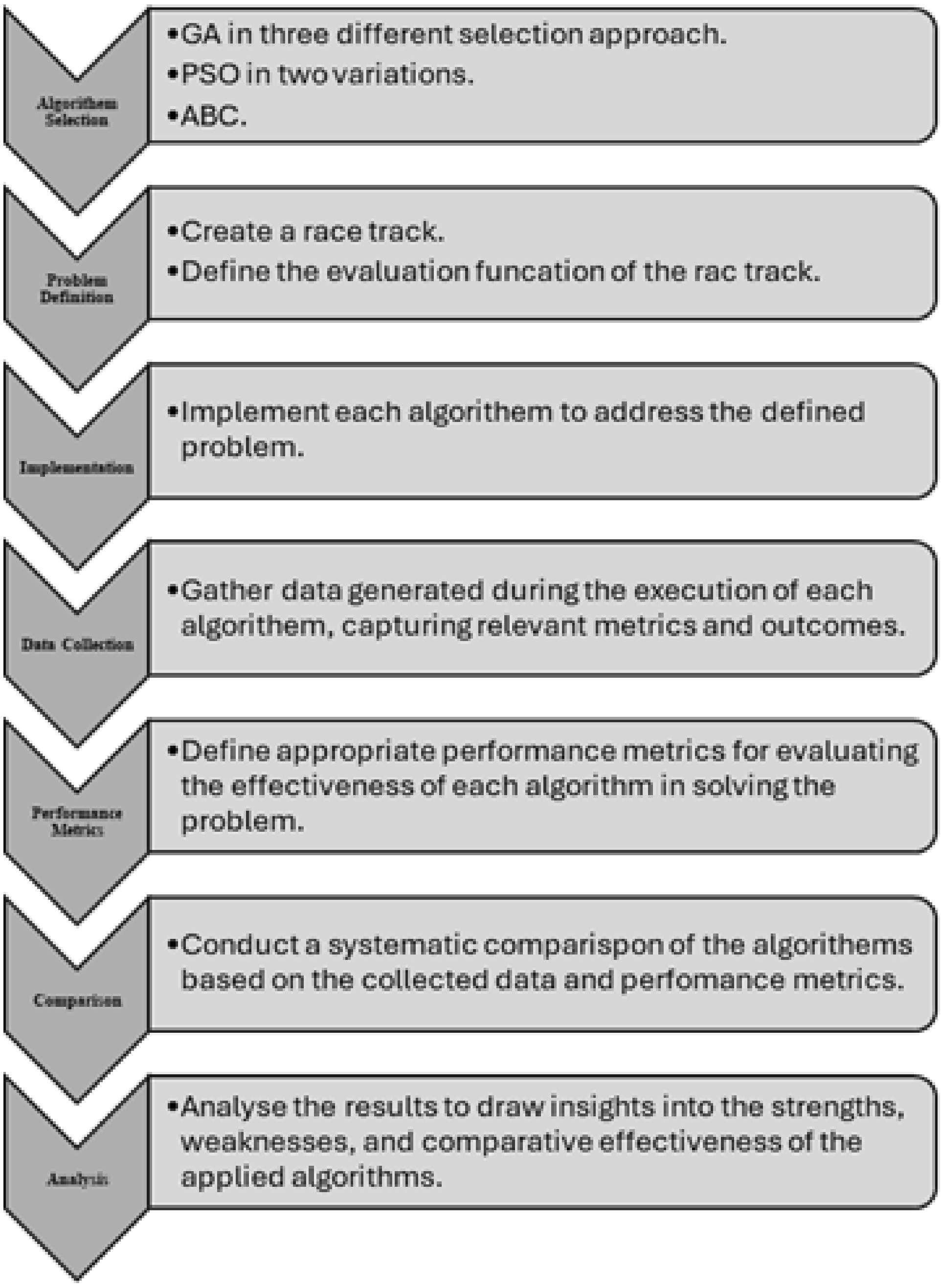
In recognition of the inherent stochastic nature of these algorithms, in this study we addressed potential performance variations by considering the average of the performance over 10 runs for each algorithm. This multi-run approach aims to provide a more robust understanding of the algorithm performance, smoothing out stochastic fluctuations and contributing to a more reliable comparative analysis.

In summary, the research method involved systematically applying the selected algorithms to a defined problem, creating a race track, and using relevant analyses, as illustrated in Figure 3. The experimental design, characterized by consistent variables and a focus on algorithmic differences, aimed to yield insights into the comparative effectiveness of the algorithms in addressing the research problem.

PCG Task

Our research focused on evaluating algorithms that generate unique race tracks for racing games. The first step involved creating a population of 600 tracks. This is achieved by generating random tracks with specific characteristics. These tracks were designed as closed loops centered around a central point. Like the points on a circle, each point on the track has a distance (radius) from this central point, as illustrated in Figure 4. This figure highlights the importance of calculating the coordinates of each point using polar coordinates.

Figure 3. Research design



- GA in three different selection approach.
- PSO in two variations.
- ABC.

- Create a race track.
- Define the evaluation funcation of the rac track.

- Implement each algorithem to address the defined problem.

- Gather data generated during the execution of each algorithem, capturing relevant metrics and outcomes.

- Define appropriate performance metrics for evaluating the effectiveness of each algorithm in solving the problem.

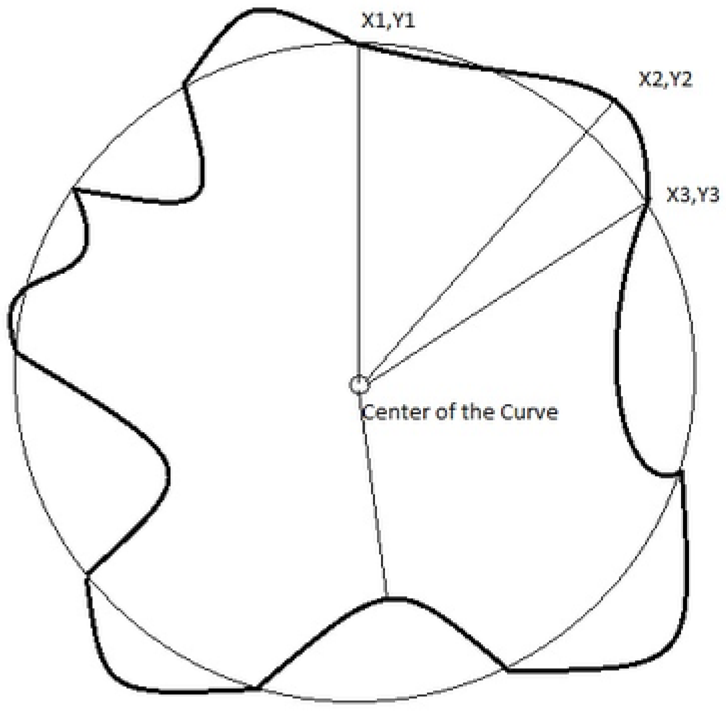
- Conduct a systematic comparispon of the algorithems based on the collected data and perfomance metrics.

- Analyse the results to draw insights into the strengths, weaknesses, and comparative effectiveness of the applied algorithms.

Note. GA = genetic algorithm; PSO = particle swarm optimization; ABC = artificial bee colony.

As illustrated in Figure 4, points such as (X1, Y1), (X2, Y2), and (X3, Y3) represent the connection points. Polar coordinates, which specify a point's location on the basis of the distance from a central point (radius) and the angular direction, are ideal for generating these initial points.

Figure 4. Visualization of a race track segment construction using polar coordinates



We chose to create tracks with 18 segments, each contributing to the overall curvature and the desired features of the track. To construct these 18 segments, we first determined the connection points between them. These connection points are crucial to ensure a continuous and well-formed race track. We achieved this by initially generating them using polar coordinates.

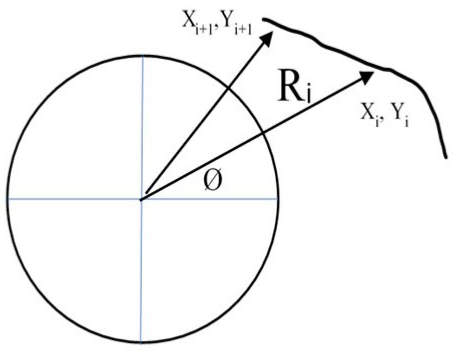
For our final track layout, we needed connection points in terms of their x and y (Cartesian) coordinates. Therefore, the initially generated polar coordinates were converted into their corresponding x- and y-axis values. We developed an efficient track creation method that uses the inherent advantages of polar coordinates to produce initial points. This process begins at the origin and represents the center of the circular track under consideration. To introduce an element of randomness and variation, we selected a set of 18 radial distances, R_i, for these initial points. R_i represents the distance of each point from the central point.

The polar coordinates are appropriate for this first phase because of their representational efficiency. In a polar coordinate system, a point's location is defined by two key parameters: (a) the radial distance R_i and (b) angular position ϕ. The radial distance signifies the distance of the point from a designated centered point (the origin), and the angular position ϕ represents the angle formed by a line connecting the point and the origin with the positive x-axis. These parameters are mathematically denoted by (R and ϕ). In contrast, the Cartesian coordinate system defines the location of a point based on its horizontal (x) and vertical (y) distances relative to the origin. as shown in Figure 5.

To convert these initial polar coordinates (distance and angle) into the final Cartesian coordinates (x and y) used for the track layout, we used well-established standard polar-to-Cartesian conversion, as in Equations 1 and 2.

$$X_i = R_i \times \mathrm{COS}(\varnothing_i). \tag{1}$$

Figure 5. Generation of different points using a decreasing angle (φ) for each round



$$X_i = R_i \times \text{SIN}(\varnothing_i), \tag{2}$$

where x and y represent the desired Cartesian coordinates, R_i denotes the radial distance in the polar system, ϕ denotes the angle (typically in radians), COS represents the cosine function, and SIN represents the sine function.

The significance of these conversion equations lies in their ability to transform the coordinates from the polar system (characterized by distance and angle) into the Cartesian system (characterized by horizontal and vertical distances). This transformation is crucial for defining the final track points in a game environment; specifically, the radial distance R_i for each point (i) was randomly generated within a range of 150–350 pixels from the center. This introduces a crucial element of randomness in the track layout. We chose 20° reduction because we had 18 segments. Dividing 360° (full circle) by 18 points resulted in a 20° separation between each point.

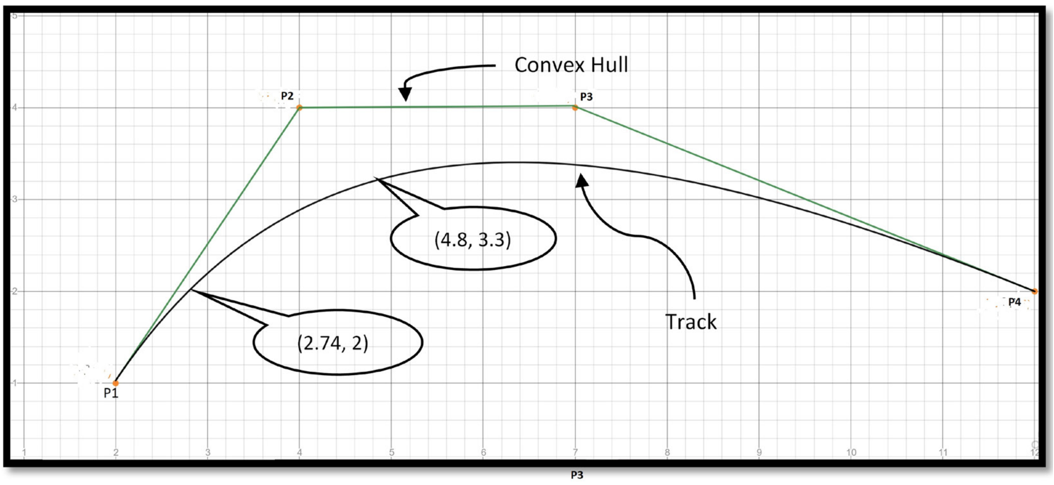
In additi8on, the polar angle ϕ is systematically reduced by 20° for each subsequent point (i + 1) compared to the previous point (i). This reduction started at 20° and continued until it reached a full circle (360°). As mentioned earlier, the curve has a central point, and this reduction contributes to the smooth curvature of the resulting race track.

Following the generation of these 18 key points using polar coordinates, as described earlier, they are seamlessly interconnected with each two neighbor points using Bézier curves (Farin, 2006). This mathematical approach allows the construction of a closed loop representing the final race track.

Bézier curves, the cornerstone of computer graphics, offer a powerful tool for defining smooth and visually appealing paths. Unlike lines or circles with a fixed curvature, Bézier curves allow the creation of more dynamic and organic shapes. This characteristic makes them ideal for generating realistic and engaging race tracks. These curves operate on the basis of a set of control points that act as invisible guides that influence the overall shape and curvature of the resulting path. In our case, we specifically used cubic Bézier curves, which require four control points per track segment. These control points are denoted in Figure 6 by P1, P2, P3, and P4. The roles of each control point in shaping the Bezier curve segment are as follows:

- P1 and P4: These control points directly correspond to the two connection points generated earlier using polar coordinates. These represent the starting and ending points of each track segment, respectively.
- P2 and P3: These are the two additional control points strategically chosen for each segment. They do not necessarily lie on the final curve, but they significantly influence its shape.

Figure 6. Generating different track shapes using the cubic Bézier method



Although the concept of convex hulls can be relevant to Bézier curves in some advanced applications, in particular when dealing with complex shapes or collision detection, it is not directly applicable to our current explanation of control points. Convex hulls represent the outermost boundary of a set of points, and their use with Bézier curves typically involves ensuring that control points are positioned within the desired shape. By understanding the influence of each control point, one can effectively manipulate the curvature and overall shape of individual track segments. This allowed us to generate a diverse range of race tracks, each offering a unique driving experience within the game environment.

This variation in the control point positions (P2 and P3) significantly contributes to the observed variability in the individual track segments. By introducing this randomness, we ensured that the generated tracks exhibited a range of shapes and curvatures, thereby enhancing the overall dynamism and visual appeal of race tracks.

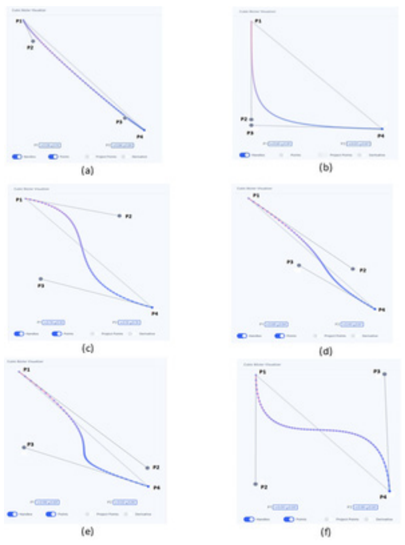
Evaluation (Fitness) Function

This study assumed that some track shapes are more desirable than others, although the intention of this research was not specifically to determine whether the algorithms can find an optimal race track. Instead, the focus was on comparing the performance of the algorithms against a given fitness function, although the fitness function itself may not be able to produce a "perfect" race track. Therefore, the fitness function we used in this study was relatively simple and aimed to direct the algorithms toward finding tracks with curved shapes in their segments, which could be considered preferable to almost straight tracks. This is similar to the work of Prasetya and Maulidevi (2016), who considered both the curvature and speed profile as factors in optimizing race tracks.

Because the intent of this work was to compare the algorithm performance and not find an optimal track, we simplified it to focus only on curvature on the assumption that the track is made more challenging by being more curved. Therefore, the fitness function scales the extent to which the given segment is rounded.

After studying different curve samples generated by the cubic Bézier function (Heckel, 2021), we observed how the curve shape depended on the two control points (P2 and P3) and their distances from the segment edges (P1, P4), as shown in Figure 7. An examination of this figure reveals that the curve samples in Panels (b), (c), (e), and (f) present some challenges. In contrast, the samples in

Figure 7. Different samples of Bézier curve generation



Panels (a) and (d) appear like straight lines. This demonstrates why the samples in Panels (b), (c), (e), and (f) were preferable for creating curved shapes.

Our analysis revealed that the distance between a control point and its neighboring segment edges significantly affects the scale of the resulting curve. On the basis of this observation, we developed a method that uses this distance to determine the weight of each segment. This ensured that each segment effectively contributed to the overall shape of the track.

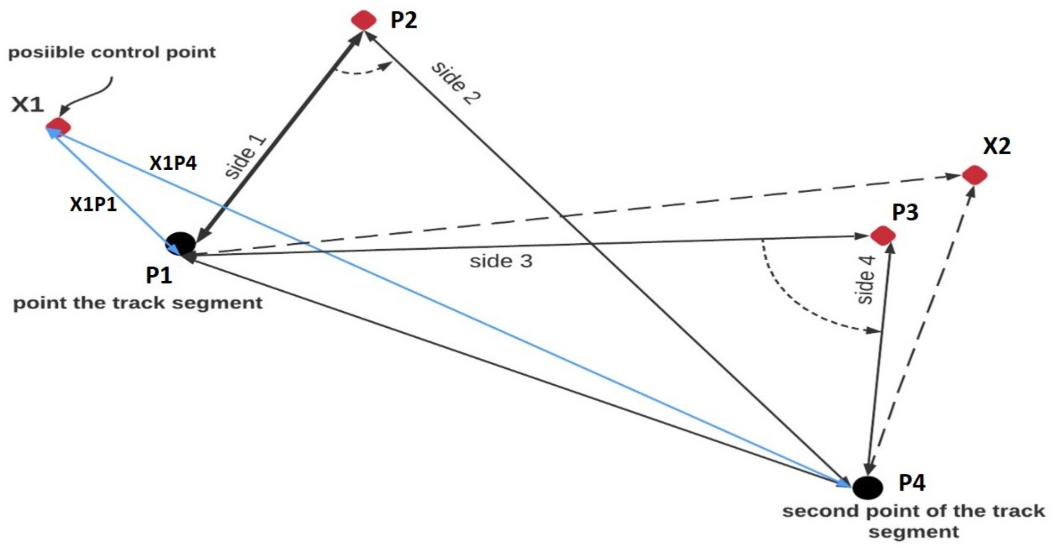
Figure 8 illustrates the process for determining the weights assigned to each track segment. This depends on the positions of the control points in the calculation. P1 and P4 represent the two edges of a segment, whereas P2 and P3 are control points. In the example shown, the specific positions of P2 and P3 are likely to generate a good curve using the Bézier method. In contrast, the control points labeled X1 and X2 represent positions that might generate undesirable curves.

We calculated the distance between control point P2 and the first edge point P1 of the segment, denoted in Figure 8 as side 1. We then calculated the distance between P2 and the next edge point, P4, denoted as side 2. Considering X1 as another possible position for a control point, we followed the same approach, calculating the distances between X1 and both edges P1 and P4.

Our analysis revealed that the difference between side 1 and side 2 was smaller than that between X1P1 and X1P4. We performed the same comparison for P3 and X2 to gain a better understanding of the ideal control-point positions. This observation played a crucial role in the development of our weight calculation formula. We used the difference between these distances (subtraction results) for both control points (P2 and P3) or in cases with potentially bad curves (X1 and X2). It is important to note that the weight assigned to each segment was calculated by summing the difference values obtained for both control points (P2 and P3) or, in negative scenarios, X1 and X2.

Mathematically, the subtraction results of $\left\| \overrightarrow{P1P2} - \overrightarrow{P2P4} \right\|$ is less than that of $\left\| \overrightarrow{P1X1} - \overrightarrow{P4X1} \right\|$. Therefore, P2 is a better position than X1 in terms of making the curve more rounded. The same approach can also be applied to P3. The conclusion comes from the summation of the two results to reflect the impact of both control points. In summary, the fitness function is a mathematical formula that generates a numerical scale for each track segment W; see Equation 3.

Figure 8. Illustration of the numerical scale of the track segment



$$W_i = \left\| \overrightarrow{P_iP_{i2}} - \overrightarrow{P_{i+1}P_{i2}} \right\| + \left\| \overrightarrow{P_iP_{i2}} - \overrightarrow{P_iP_{i2}} \right\|, \tag{3}$$

where i represents the segment number, P_i denotes the endpoints of the segment, and P_{i2} $andP_{i3}$ are the control points for segment i.

The summation of these results is the total weight of the track, as shown in Equation 4. A track with a small total weight is preferable.

$$W_{track} = \sum_{i=0}^{17} W_i. \tag{4}$$

Metaheuristic Implementation

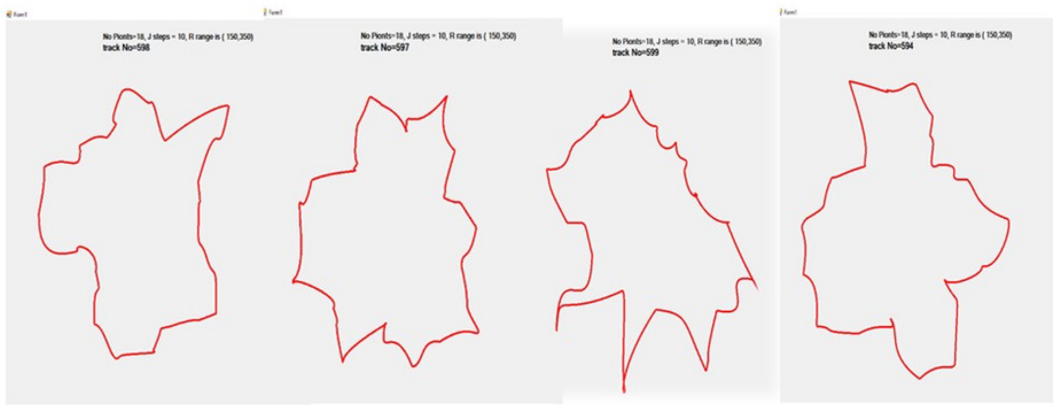
The initial stage of the study implemented three metaheuristic approaches (GA, ABC, and PSO) and used these algorithms to generate race tracks to gain knowledge of their relative performance in producing game levels.

The performance is determined in two dimensions: (a) the effectiveness of the algorithms in terms of their ability to converge on a better solution in a fixed number of iterations and (b) the consistency of this effectiveness across multiple attempts to generate race tracks. Figure 9 shows some typical tracks generated in this study; note that sharp changes in direction in each would not be desirable in a track that was intended to be playable. However, this does not detract from the comparison of the algorithm performance and can be addressed in future work by using an improved fitness function.

GA Implementation

The GA was the first metaheuristic approach used in this study. It consisted of four steps after the initial population was generated. The first step involved selecting a pair of tracks (parents). However, the selection process used three different approaches: (a) a biased roulette wheel, (b) a tournament, and (c) a combination of the two approaches. The second step was the evaluation, which depends on the weight of the track. The track with the lowest weight was most likely to be considered. The third step

Figure 9. Sample of tracks from the initial population


No Pionts=18, J steps = 10, R range is (150,350)
track No=598
No Pionts=18, J steps = 10, R range is (150,350)
track No=597
No Pionts=18, J steps = 10, R range is (150,350)
track No=599
No Pionts=18, J steps = 10, R range is (150,350)
track No=594

was the reproduction process, in which individuals' tracks pass their segments to the next generation. The last step was mutation, in which newly formed children are subjected to transformation. The genes were randomly selected for modification by exploring new values. After the initial population is generated, the steps described in the following four sections are taken.

1. **Selection of the Parents.** Our approach to parental selection within the GA lever used three distinct techniques: (a) biased roulette wheel selection; (b) tournament selection; and (c) a combined approach that utilizes both methods with equal probability (50%), denoted as "Mix." Biased roulette wheel selection represents a variation of the standard roulette wheel selection technique used in GAs (Katoch et al., 2021).

In this method, a virtual wheel is divided into sections (pie slices) proportional to the fitness scores of the individual tracks within the population. This ensures that tracks exhibiting superior performance (lower fitness scores) occupy larger sections of the wheel, and the selection process commences with the rotation of the virtual roulette wheel. A fixed point on the wheel perimeter serves as a selection marker. The first track segment, whose corresponding section reaches the marker, is selected as the first parent. This process is repeated to select a second parent.

The inherent advantage of biased roulette wheel selection is its ability to probabilistically favor individuals with higher fitness scores. Because these superior tracks occupy larger sections on the wheel they have a greater chance of aligning with the selection marker during rotation. Thus, this approach promotes the propagation of desirable track characteristics to the next generation, accelerating the evolutionary process toward optimal track designs. In the implementation, we used the following steps:

1. Calculate the total fitness
2. Calculate the proportion of each individual P_i according to Equation 5

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i}, \tag{5}$$

where n is the number of tracks in our population and f_i is the fitness value of each track.

3. Calculate the cumulative proportion CP_j of each track according to Equation 6

$$CP_j = CP_{j-1} + P_j. \tag{6}$$

4. Generate a random number R between 0.0 and 1.0
5. Starting from the top of the population, the individual for which CP_j goes above R is the selected track (parent).
6. The same process is followed when selecting the other parent.

The second selection strategy used in our GA framework is the well-established tournament-selection method. This technique is widely used to identify the most suitable individuals (tracks) from the current generation by means of a competitive selection process (Katoch et al., 2021). Tournament selection involves creating a group that consists of a predefined number (K) of individuals from the population. These individuals were chosen randomly. In our implementation, we opted for a K value of 2, essentially creating pairwise competition.

After random selection, competition is conducted between K individuals. The fitness score, which serves as a measure of an individual's performance (track quality), was used as the primary criterion for this competition. The individual with the lower fitness score (better performing track) is designated as the winner and progresses to the next generation. This tournament selection process is iterated multiple times, allowing for gradual identification and promotion of the most promising candidates (optimal tracks) to the next generation. This iterative approach facilitates the convergence of the GA toward populations comprising superior track designs.

Despite the previous selection methods, we incorporated into our third approach a hybrid strategy that combined biased roulette wheel selection and tournament selection. During each iteration, we generated random numbers between 0 and 100 to determine the selection method. If the generated number was less than 50, we used the biased roulette wheel method; otherwise, we used the tournament method. This dynamic approach enables adaptability and flexibility in selecting the most suitable strategy for each iteration.

2. **The Evaluation Process.** The second step is the evaluation, which is based on the weights assigned to each track. The track with the lowest weight is more likely to be considered for further processing and potential improvement.
3. **The Reproduction Step.** The third step is the reproduction process, in which the tracks of individuals are passed on to the next generation. This process includes a crucial phase known as the *crossover point*. The crossover point is selected at half the number of genes and determines how the segments (genes) of the selected parent tracks are swapped to generate offspring. This step facilitates the exploration of different genetic combinations, as shown in Figure 10.

4. **The Mutation Step.** Mutation serves as the final step in the GA algorithm. In this crucial stage, specified genes (referred to as *segments* of the track) within the newly formed offspring are transformed. These genes are selected at random for modification, enabling the exploration of new values and potential enhancements in the genetic diversity of the population. The assigned mutation rate of 0.05 indicates the proportion of genes that are subject to transformation. We chose this rate after conducting different experiments and found it to be the best.

By iteratively performing these steps, the GA aims to optimize and evolve the population, leading to improved solutions over time. These steps provide an overview of the GA process used in our research methodology. The actual implementation on the CSharp platform involves additional

Figure 10. Genetic algorithm crossover

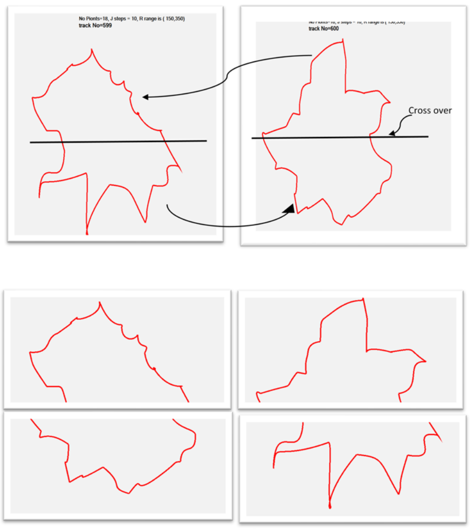
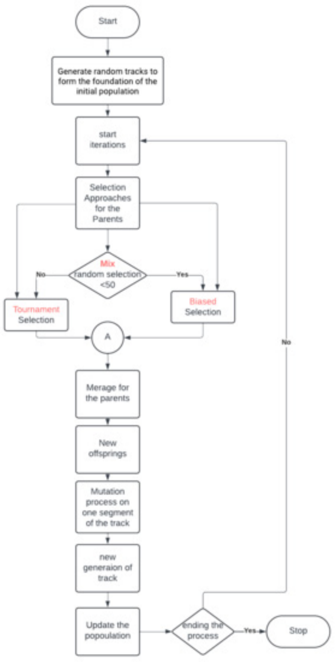


Figure 11. Steps of genetic algorithm approach in track generation



details and specific parameters based on the research objectives and the problem domain, as shown in Figure 11.

ABC Implementation

Our investigation extended beyond GAs to also explore the potential of the ABC algorithm. This metaheuristic approach draws inspiration from animal behavior and the principles of SI (Karaboga et al., 2014). SI techniques are characterized by their reliance on a collective of individuals, often referred to as *workers*, *particles*, or *agents*. These individuals collaborate and interact with each other to collectively search for optimal or near-optimal solutions for complex problems (Karaboga et al., 2014).

The ABC algorithm specifically mimics the foraging behavior observed in honeybee colonies. As honeybees collectively locate food sources with a high nectar yield, the algorithm uses a population of artificial bees to explore potential solutions—track designs, in our case. Self-organization and division of labor, which are fundamental principles of SI (Alyaseri & Aljanaby, 2014; Dedeturk et al., 2021) are evident in natural honeybee colonies. These principles are further complemented by the satisfaction principles of the ABC algorithm.

Although not as ubiquitous as GAs, the ABC algorithm remains a popular choice for solving diverse optimization problems. In this study we used the ABC algorithm to iteratively generate an optimized race track through a structured four-phase process:

1. *Initial population phase:* During the initial population phase, 600 unique race tracks were randomly generated using the methods described earlier. Each of these tracks, denoted by χi, represents a potential food source location within the ABC algorithm.
2. *Employee bee phase*: The second phase involves employed bees actively modifying the existing race tracks. These bees evaluate the quality (fitness) of their assigned tracks and explore the search space by applying local search techniques, such as mutations. During each update, a mutation rate of 0.05 determines the proportion of segments within a track that will be randomly modified. This allows bees to adapt and refine their solutions within the evolving landscape of the potential race tracks.
3. *Onlooker bee phase*: The onlooker bee phase involves the selection of promising areas within the search space. Here, onlooker bees evaluate the solutions (race tracks) presented by the employed bees along with their corresponding fitness values. They favor tracks with higher fitness (lower values typically indicate better tracks). This selection process is influenced by probability $P\ (\chi i)$ in the best direction, as defined in Equation 7. Considering this probability, onlooker bees aim to discover new, potentially improved tracks. This phase emphasizes exploration and ultimately aims to enhance the overall quality of race tracks through the iterative selection of superior solutions.

$$P(x_i) = \frac{f_i}{Maxf}, \tag{7}$$

wheref_i is the fitness value and *Maxf* is the maximum fitness in the population.

4. *Scout bee phase*: The final stage in the ABC method is the scout bee phase. Employed bees that failed to improve their assigned solutions after a certain number of attempts became scout bees. Unlike employed bees, scout bees abandon their current solutions and randomly generate entirely new race tracks. This introduces new possibilities for exploration by venturing into uncharted territories within the search space. This random exploration allows the algorithm to expand its search boundaries and potentially discover superior solutions. By introducing new possibilities and exploring unexamined areas, scout bees can contribute to the ongoing search for optimal race tracks within the solution space.

In summary, the ABC algorithm progresses through individual phases, starting with the generation of a diverse initial population of race tracks. The subsequent phases involve the dynamic adjustment of race track positions by employee bees guided by probabilities, strategic selection of promising race tracks by onlooker bees, and exploration of new possibilities by scout bees. This systematic approach ensures a comprehensive search for optimized race tracks within the solution space.

PSO Implementation

The PSO method was the last algorithm under study in this research. Unlike GA, PSO operates without traditional modification operators such as crossover or mutation. In PSO, particles are considered potential solutions for navigating the problem space by following the best positions of the current best particles. Of note, PSO has demonstrated computational efficiency, requiring fewer computations than GAs for problem-solving (Lima et al., 2011). It is an efficient algorithm in terms of both memory utilization and computational speed (Yusup et al., 2012).

PSO draws inspiration from emulating the optimization behavior of a flock of birds in solving complex mathematical problems. Visualize a swarm of birds that collectively determine a landing site while flying over an area. This decision-making process optimizes food supply and minimizes the risk of predators. In this context, bird movements resemble choreography; they fly in unison until they indicate the ideal landing place, and the entire flock lands simultaneously (Shami et al., 2022). The classical version of PSO determines a variable represented by a vector **X** as shown in Equation 8.

$$\mathbf{X} = \left[\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots\ldots, \mathbf{X}_n\right]. \tag{8}$$

Vector **X** represents the initial population of 600 race tracks ($n = 600$ in this case). The goal is to minimize the value of $f(\mathbf{X})$, which indicates a better performance based on the chosen evaluation criteria for race track quality. The fitness function, $f(\mathbf{X})$, evaluates a race track's quality on the basis of various survival criteria.

In a swarm of P particles, each particle has a position vector $\mathbf{X}_i^t$ (Equation 9) and a population velocity vector $\mathbf{V}_i^t$ (Equation 10) at iteration T.

$$\mathbf{X}_i^t = \left(x_{i1,}\, x_{i2}, x_{i3}, \ldots\ldots\ldots x_{in}\right)^T. \tag{9}$$

$$\mathbf{V}_i^t = \left(v_{i1,}\, v_{i2}, v, \ldots\ldots\ldots v_{in}\right)^T. \tag{10}$$

These vectors are updated across the dimension j, which corresponds to the number of segments in each track. This update is governed by Equations 11 and 12, where $i = 1, 2, 3, \ldots, P$ and $j = 1, 2, 3, \ldots, n$. In this context, each segment was updated on the basis of the information obtained from Equation 11.

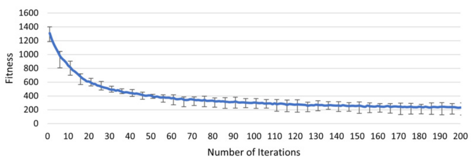
The acceleration coefficient *C1* dictates the extent to which information is extracted from the best segment in the track so far in the iteration, whereas C_2 determines the global information of the population by examining the best track in the current iteration to enhance the current track; therefore, the improvement applied for each segment in the track is shown in Equation 12. However, in our study, C_1 is assigned a value of 2 for both versions, C_2 are assigned a value of 4 for PSO#1 and a value of 5 for PSO#2.

$$\mathbf{V}_{ij}^{t+1} = \omega \mathbf{V}_{ij}^t + C_1 r_1^t\left(pbest_{ij} - \mathbf{X}_{ij}^t\right) C_2 r_2^t\left(gbest_{ij} - \mathbf{X}_{ij}^t\right). \tag{11}$$

In Equation 11, the parameters are defined as follows:

- ω is the inertia weight;

Figure 12. Genetic algorithm average convergence (mix)



- $\mathbf{V}_{ij}^{t+1}$ is the updated velocity of track i in segment j at iteration $t + 1$;
- $\mathbf{V}_{ij}^{t}$ is the current velocity of track i in segment j at time t;
- C_1 and C_2 are acceleration coefficients, determining the influence of personal best (best segment; *pbest*) and global best (*gbest*) information on the best track in the population; and
- r_1^t and r_2^t are random values between 0 and 1 generated at each iteration to introduce stochasticity.

$$\mathbf{VX}_{ij}^{t+1} = \mathbf{X}_{ij}^{t} + \mathbf{V}_{ij}^{t+1}. \tag{12}$$

Choosing an appropriate value for ω is crucial for the performance of the PSO algorithm. The typical range of ω is between 0 and 1, and in our case is 0.3. However, the selection of ω may depend on the specific optimization problem and the behavior we want the algorithm to exhibit.

RESULTS

Our aim was to compare the performance of three different metaheuristic search algorithms on a PCG task, specifically, the generation of tracks for a racing game. We evaluated each selected metaheuristic method in terms of its efficiency and effectiveness. Usually, the fitness value of the tracks drops between 1,600 and 100 points, depending on the scale considered in this study, as described in Equations 3 and 4.

The maximum iteration number (200) and population size (600) are the same to permit comparisons of the performance of all methods equitably; however, the lack of convergence criteria is noted. As in many real-world applications, there may be benefits in running the algorithms until convergence is detected. In addition, each method was examined 10 times. We took into consideration the average of their outputs for each algorithm. We also evaluated the best values, mean values, worst values, and standard deviations.

GA Application Results

The application of the GA is discussed in the “GA Implementation” section, and the results presented here relate to the performance of the three different selection approaches that we implemented. Figure 12 shows the improvement in the fitness function over the 200 generations for which the algorithm was run using the mixed selection approach. In the figure, the line indicates the average fitness for each generation across 10 runs of the algorithm. The error bars indicate the variation in individual runs relative to the average.

The convergence of the algorithm is typical of evolutionary approaches, showing a relatively quick initial convergence in the first 50 generations and then a gradual improvement over the remaining generations. The initial randomly generated tracks in the first generation have fitness values in the 1,200–1,400 range, which is fairly consistent with all the results in this study, and the final tracks have an average fitness of approximately 200, although with a relatively large variation around that value for each of the individual runs.

Figure 13. Genetic algorithm average convergence (biased)

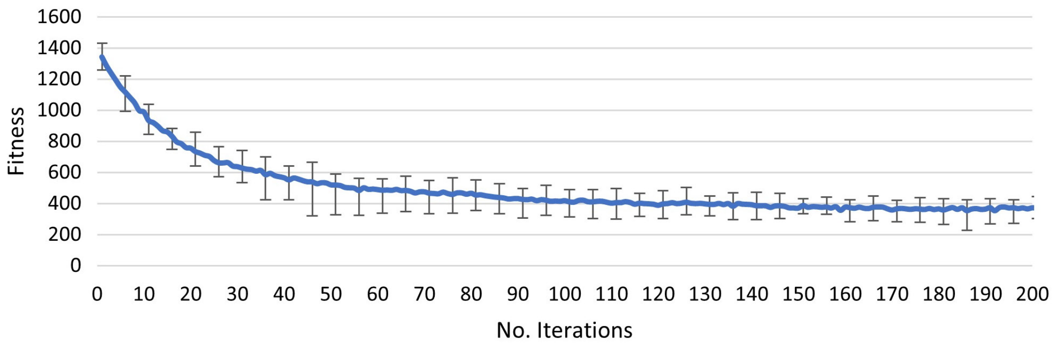


Figure 14. Genetic algorithm average convergence (tournament)

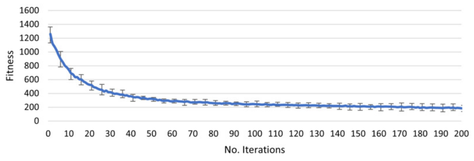


Figure 13 displays the results of using GA with a biased selection method. Although the initial average fitness is comparable to that in Figure 12, the initial convergence is slightly slower and the final fitness values are higher, with an average value of approximately 400.

Figure 14 shows the results of using the tournament selection approach, where the average initial fitness is again comparable with the previously presented results. However, in this case the initial convergence was much faster. Considering the number of generations required to reach the final average fitness achieved using the biased selection approach, tournament selection reaches a fitness value of 400 after an average of 30 generations, compared with the 50 generations required using the mix selection approach. Although the final average fitness shows no real difference from that achieved using the mixed selection method, the variability across multiple runs is significantly lower.

ABC Application Results

Figure 15 depicts the results of multiple runs of the ABC algorithm, which shows a marked difference from any of the GA results. The initial speed of convergence is much slower, and although no attempt has been made to quantify this, if convergence were to be considered a form of exponential decay, then the lambda value of the curve would be much lower than that of GA convergence. The initial average fitness value is marginally higher than the results for the different GA implementations, and the convergence is slower, reaching the previously indicated threshold value of 400 in just over 80 iterations. However, unlike the GA implementations, the convergence is less asymptotic, and improvement continues over the entire 200 iterations, leading to a final average fitness value of more than 100 points. More important, the variability across multiple runs is virtually nonexistent, implying that this performance is consistent and that multiple runs are not required to find a good solution.

PSO Application Results

Similar to the GA implementation, two variations of the PSO algorithms were implemented with two different values of the C_2 acceleration coefficient in Equation 11, which influences the extent to which the search extracts information from the global position. For both implementations, the

Figure 15. Artificial bee colony convergence

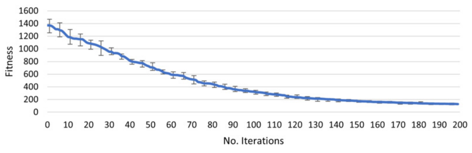


Figure 16. Particle swarm optimization convergence (PSO#1)

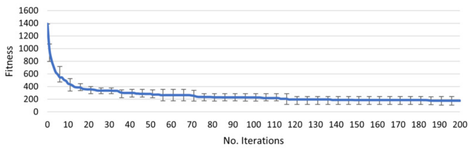
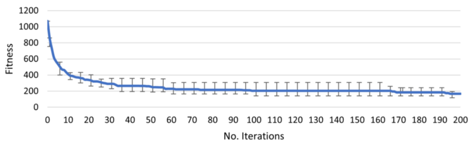


Figure 17. Particle swarm optimization convergence (PSO#2)



value assigned to C_1 was 2. The first implementation, denoted as *PSO#1*, used a C_2 value of 4, and the second implementation, denoted as *PSO#2*, used a C_2 value of 5. Figure 16 depicts the results of the first implementation (PSO#1). Compared with both the GA and ABC results, PSO shows a very fast initial convergence, reaching the threshold fitness value of 400 points in just 10 iterations. The convergence curve then becomes almost asymptotic but shows slow and steady improvement over the remaining iterations, leading to a final average fitness of less than 200. Overall, the final fitness quality and variability around the average are comparable to the best results achieved using a GA.

Figure 17 presents the results for the second PSO implementation (PSO#2), with a higher emphasis on global position information. These results show little significant variation in the convergence behavior, reaching the 400 threshold value in slightly over 10 iterations and only a marginal improvement in the final average fitness function value and variability. It is worth noting that these results arise from an initial population fitness that is slightly lower than the other results, which may have influenced the overall convergence.

Metaheuristics Results Comparison

The results presented in Figures 12–17 provide a visual representation of the performance of the algorithms and an initial indication of their relative merits. Table 1 presents more information regarding the quality of the solutions obtained for each algorithm across multiple runs. These data essentially correspond to the variation observed in each figure at the 200 iteration mark.

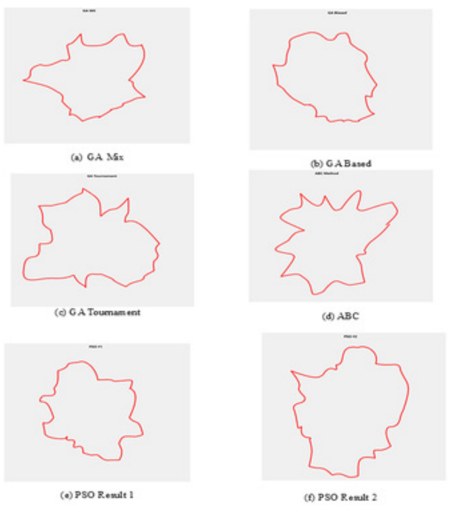
In relation to the best solution found by each algorithm across the 10 runs, the overall best solution was found by PSO#1, with a fitness value of 106.88. Comparing absolute values has little meaning in the context of the fitness function because there is no real means of determining whether a fitness value of 115.86 results in a significantly different quality of race tracks. However, there is a relatively clear difference between the best fitness values resulting from both the ABC and PSO algorithms, in particular when compared with the biased selection implementation of the GA.

Table 1. Comparison of different algorithms

Metric	GA(Mix)	GA(Biased)	GA(Tournament)	ABC	PSO#1	PSO#2
Worst	299.47	443.28	225.63	138.67	243.64	194.60
Best	121.75	304.30	137.74	113.98	106.88	115.86
M	233.09	370.10	182.90	125.90	175.50	166.40
SD	52.35	46.13	25.28	8.73	38.92	23.17

Note. GA = genetic algorithm; ABC = artificial bee colony; PSO = particle swarm optimization.

Figure 18. Comparison of resulting tracks



Note. GA = genetic algorithm; ABC = artificial bee colony; PSO = particle swarm optimization.

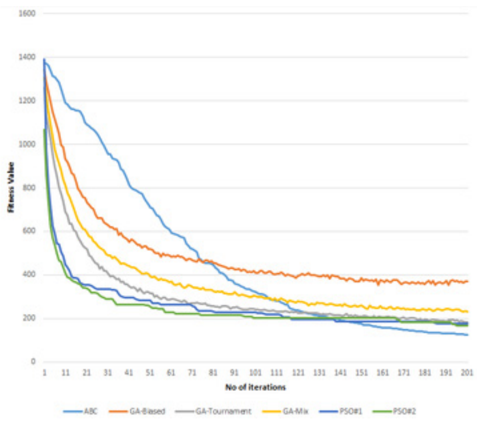
Although the aim of this study was not to attempt to determine an optimal race track, it is worth considering the output of each algorithm in terms of the race tracks produced. Figure 18 shows the best tracks produced by applying the GA, ABC, and PSO algorithms.

A comparison of the ABC solution with the solution produced by GA using biased selection reveals a clear difference in terms of track curvature, with the ABC algorithm being significantly more curved. Race tracks clearly are not necessarily suitable for use in a game because of abrupt changes in direction; however, the intention of this research was not to produce the optimal race track. The fitness function appears to drive the solution to a specified goal even though the goal may be flawed. This confirms that the insights derived from these results can be applied to scenarios with improved fitness functions.

Figure 19 provides further insight into the performance of the algorithms. Although ABC achieved the best results in terms of the mean value of fitness, it also exhibited the slowest convergence speed. At first, the ABC algorithm showed slower convergence, but it significantly improved later and maintained a consistent performance with minimal variability.

It is important to note that, on the basis of their raw values, both ABC and PSO appear competitive. Although PSO might have a slight advantage in quickly reaching the absolute best solution, ABC tends to achieve a better average fitness over time. Further studies are required to determine their strengths and weaknesses definitively. Interestingly, both ABC and PSO belong to the SI algorithm

Figure 19. All the results for all the algorithms based on the average fitness



Note. ABC = artificial bee colony; GA = genetic algorithm; PSO = particle swarm optimization.

category. This common classification suggests that this type of algorithm is well suited to problems related to PCG.

DISCUSSION AND FUTURE DIRECTIONS

In this study, we investigated the performance of various computational algorithms for generating race tracks for video games. Our primary focus was to challenge the prevalent reliance on GAs by exploring alternative methods. Our findings show that GAs' selection strategy significantly affects their effectiveness. Selection methods that prioritize rapid convergence might initially identify suitable tracks but may limit the exploration of the design space, potentially leading to repetitive layouts. Conversely, methods that promote broader exploration can lead to slower convergence but potentially discover more diverse and engaging track layouts. Future work could explore the systematic evaluation of different selection strategies tailored to race track generation with GAs.

Compared to GAs, PSO offers a faster convergence rate, making it advantageous in scenarios with limited computational resources or real-time content generation requirements. However, the optimal performance relies on careful tuning of the inertia weight parameter. A higher initial value encourages exploration, allowing particles to explore a wider range of potential track layouts. As the optimization process progresses, gradually reducing this parameter focuses on the search for promising regions identified by successful particles. Future research could investigate adaptive mechanisms that dynamically adjust this parameter based on observed convergence behavior.

The ABC algorithm exhibits a high degree of repeatability in generating high-quality tracks. This consistency might be valuable for applications in which reliable and consistent track generation is a priority; however, ABC's initial convergence tends to be slower than that of PSO. Future studies could explore hybrid approaches that combine the strengths of ABC with other algorithms. For example, an initial exploration phase using PSO followed by exploitation with ABC can leverage the benefits of both, achieving faster initial convergence while maintaining ABC's high repeatability.

It is important to understand that there is no "best" algorithm overall, and the selection of an algorithm involves trading off the characteristics of the algorithm and matching them to a given problem. The findings of this study suggest that game developers using PCG for race tracks should explore alternatives to GAs. The optimal choice depends on the trade-off between the factors. For scenarios with limited computational resources or real-time generation needs, PSO offers faster convergence, whereas for high- quality, diverse, and engaging tracks the ABC algorithm or a hybrid approach that leverages its strengths might be more suitable.

This study has several limitations. Although a range of comparative studies exist in other domains, similar investigations are currently limited to PCG for video games. This study highlights the potential of alternative algorithms with different performance characteristics, warranting further exploration. Our current evaluation focused on a single metric (curvature), but future work should incorporate additional metrics, such as computational efficiency, scalability with increasing track complexity, and user studies for subjective assessments. Validation efforts will involve comparing the generated tracks with real-world designs and incorporating expert evaluations from racing game designers or professional drivers.

In addition, the current fitness function will be expanded to consider a broader range of design aspects through penalties or rewards for monotonous sections, environmental obstacles, and aesthetic qualities. Future studies should also consider more systematic investigations of different population sizes and their effect on algorithm performance; use the same number of evaluations for all algorithms, to ensure fair comparisons; and incorporate statistical tests, such as analysis of variance, to determine the statistical significance of observed differences.

This study provides insights into the performance of various algorithms for race track generation in video games. We highlight the importance of considering alternatives beyond GAs and the need for further exploration of evaluation metrics, validation procedures, and more intricate fitness functions to achieve optimal results in PCG for race tracks.

CONCLUSION

In this study, we measured and compared efficacious algorithms and their achievements in PCG tasks in various settings to determine the best race track, specifically, we evaluated GA, ABC, and PSO. We also tested GA using different selection approaches, along with PSO, in two different settings, and we developed a mathematical method that assumes an evaluation function to select the best track. The results were significant in showing that other metaheuristic approaches could considerably change search-based PCG. This study contributes to the literature by expanding our knowledge of how different algorithms are suited to PCG, clearing the way for further studies to determine the most suitable algorithms for games. This study provides a basis for developing recommendations for the selection of these methods. The goal is to ensure that the resulting games have the potential to satisfy both the needs of both the designer and the player.

CONFLICTS OF 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.

FUNDING STATEMENT

No funding was received for this work.

PROCESS DATES

07, 2024

This manuscript was initially received for consideration for the journal on 02/02/2024, revisions were received for the manuscript following the double-anonymized peer review on 07/14/2024, the manuscript was formally accepted on 05/10/2024, and the manuscript was finalized for publication on 07/16/2024

CORRESPONDING AUTHOR

Correspondence should be addressed to Sana Alyaseri; sana.alyaseri@gmail.com

REFERENCES

Abu-Mouti, F. S., & El-Hawary, M. E. (2012). Overview of artificial bee colony (ABC) algorithm and its applications. In *2012 IEEE International Systems Conference SysCon 2012* (pp. 1–6). IEEE. 10.1109/SysCon.2012.6189539

Alves, T., Coelho, J., & Nogueira, L. (2018). Content generation for massively multiplayer online games with genetic algorithms. In *Applied computational intelligence and mathematical methods: Computational methods in systems and software 2017* (*Vol. 2*, pp. 37–49). Springer. 10.1007/978-3-319-67621-0_4

Alyaseri, S., & Aljanaby, A. (2014). Population based heuristic approaches for grid job scheduling. *International Journal of Computer Applications*, *91*(5), 45–50. 10.5120/15881-4853

Alyaseri, S., Sinha, R., & Nand, P. (2024). *Systematic literature review of meta-heuristic algorithms and their application in procedural content generation in the context of computer games*. Manuscript submitted for publication.

Barriga, N. A. (2019). A short introduction to procedural content generation algorithms for videogames. *International Journal of Artificial Intelligence Tools*, *28*(2), 1930001. 10.1142/S0218213019300011

Blasco, D., Font, J., Pérez, F., & Cetina, C. (2023). Procedural content improvement of game bosses with an evolutionary algorithm. *Multimedia Tools and Applications*, *82*(7), 10277–10309. 10.1007/s11042-022-13674-6

Collette, Y., Siarry, P., & Wong, H.-I. (2000). A systematic comparison of performance of various multiple objective metaheuristics using a common set of analytical test functions. *Foundations of Computing and Decision Sciences*, *25*(4), 249–271.

Connor, A. M. (1996). *The synthesis of hybrid mechanisms using genetic algorithms*. John Moores University.

Connor, A. M., & Shea, K. (2000). A comparison of semi-deterministic and stochastic search techniques. In Parmee, I. C. (Ed.), *Evolutionary design and manufacture* (pp. 287–298). Springer., 10.1007/978-1-4471-0519-0_23

Connor, A. M., & Tilley, D. G. (2016). *A comparison of two methods applied to the optimisation of fluid power circuits*. arXiv: 1605.03667. https://doi.org//arXiv.1605.0336710.48550

Couceiro, M., Ghamisi, P., Couceiro, M., & Ghamisi, P. (2016). *Particle swarm optimization*. Springer.

Dalwani, S., & Agarwal, A. (2018). Review on classification of nature inspired approach. *International Journal of Computer & Mathematical Sciences*, *7*(2), 588–595.

de Araújo, L. J. P., Grichshenko, A., Pinheiro, R. L., Saraiva, R. D., & Gimaeva, S. (2020). Map generation and balance in the Terra Mystica board game using particle swarm and local search. In Y. Tan, Y. Shi, & M. Tuba (Eds.), *Advances in Swarm Intelligence: 11th International Conference, ICSI 2020: Lecture notes in computer science* (Vol. 12145, pp. 163–175). Springer. 10.1007/978-3-030-53956-6_15

De Carli, D. M., Bevilacqua, F., Pozzer, C. T., & d'Ornellas, M. C. (2011). A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian Symposium on Games and Digital Entertainment* (pp. 26–35). IEEE. 10.1109/SBGAMES.2011.15

de Lima, E. S., Feijó, B., & Furtado, A. L. (2022). Procedural generation of branching quests for games. *Entertainment Computing*, *43*, 100491. 10.1016/j.entcom.2022.100491

de Pontes, R. G., Gomes, H. M., & Seabra, I. S. R. (2022). Particle swarm optimization for procedural content generation in an endless platform game. *Entertainment Computing*, *43*, 100496. 10.1016/j.entcom.2022.100496

Dedeturk, B. K., Akay, B., & Karaboga, D. (2021). Artificial bee colony algorithm and its application to content filtering in digital communication. In Karbas, S., Toktas, A., & Ustun, D. (Eds.), *Nature-inspired metaheuristic algorithms for engineering optimization applications* (pp. 337–355). Springer., 10.1007/978-981-33-6773-9_15

Dhal, K. G., Das, A., Gálvez, J., Ray, S., & Das, S. (2020). An overview on nature-inspired optimization algorithms and their possible application in image processing domain. *Pattern Recognition and Image Analysis*, *30*(4), 614–631. 10.1134/S1054661820040100

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, *137*, 106040. 10.1016/j.cie.2019.106040

Farahbakhsh, A., & Kheirkhah, A. S. (2023). A new efficient genetic algorithm-Taguchi-based approach for multi-period inventory routing problem. *International Journal of Research in Industrial Engineering*, *12*(4), 397–413.

Farin, G. (2006). Class a Bézier curves. *Computer Aided Geometric Design*, *23*(7), 573–581. 10.1016/j.cagd.2006.03.004

Ferreira, L., Pereira, L., & Toledo, C. (2014). A multi-population genetic algorithm for procedural generation of levels for platform games. In *GECCO Comp '14:Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 45–46). Association for Computing Machinery. 10.1145/2598394.2598489

Fister, I.Jr, Perc, M., Ljubič, K., Kamal, S. M., Iglesias, A., & Fister, I. (2015). Particle swarm optimization for automatic creation of complex graphic characters. *Chaos, Solitons, and Fractals*, *73*, 29–35. 10.1016/j.chaos.2014.12.019

Fister, I.Jr, Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). *A brief review of nature-inspired algorithms for optimization*. arXiv: 1307.4186.

Ghaheri, A., Shoar, S., Naderan, M., & Hoseini, S. (2015). The applications of genetic algorithms in medicine. *Oman Medical Journal*, *30*(6), 406–416. 10.5001/omj.2015.8226676060

Grichshenko, A., de Araújo, L. J. P., Gimaeva, S., & Brown, J. A. (2020). Using tabu search algorithm for map generation in the Terra Mystica tabletop game. In *ISMSI '20: Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence* (pp. 119–122). Association for Computing Machinery. 10.1145/3396474.3396492

Gupta, S., Abderazek, H., Yıldız, B. S., Yildiz, A. R., Mirjalili, S., & Sait, S. M. (2021). Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. *Expert Systems with Applications*, *183*, 115351. 10.1016/j.eswa.2021.115351

Heckel, M. (2021, November 2). Cubic Bézier: From math to motion. *Maxime Heckel*. https://blog.maximeheckel.com/posts/cubic-bezier-from-math-to-motion/

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing Communications and Applications*, *9*(1), 1–22. 10.1145/2422956.2422957

Kannan, S., Slochanal, S. M. R., & Padhy, N. P. (2005). Application and comparison of metaheuristic techniques to generation expansion planning problem. *IEEE Transactions on Power Systems*, *20*(1), 466–475. 10.1109/TPWRS.2004.840451

Kanović, Ž., Bugarski, V., & Băckalić, T. (2014). Ship lock control system optimization using GA, PSO and ABC: A comparative review. *Promet (Zagreb)*, *26*(1), 23–31. 10.7307/ptt.v26i1.1475

Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132. 10.1016/j.amc.2009.03.090

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57. 10.1007/s10462-012-9328-0

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, *80*(5), 8091–8126. 10.1007/s11042-020-10139-633162782

Kim, J.-M., Kim, S.-J., & Hong, S. (2017). Players adaptive monster generation technique using genetic algorithm. *Journal of Internet Computing and Services*, *18*(2), 43–51. 10.7472/jksii.2017.18.2.43

Kim, P. H., & Crawfis, R. (2015). The quest for the perfect perfect-maze. In *2015 computer games: AI, animation, mobile, multimedia, educational and serious games (CGAMES)* (pp. 65–72). IEEE. https://www.doi.org/10.1109/CGames.2015.7272964

Korn, O., Blatz, M., Rees, A., Schaal, J., Schwind, V., & Görlich, D. (2017). Procedural content generation for game props? A study on the effects on user experience. *Computers in Entertainment*, *15*(2), 1–15. 10.1145/2974026

Kramer, O., & Kramer, O. (2017). *Genetic algorithms*. Springer.

Kraner, V., Fister, I.Jr, & Brezočnik, L. (2021). Procedural content generation of custom tower defense game using genetic algorithms. In *New technologies, development and applications* (pp. 493–503). Springer., 10.1007/978-3-030-75275-0_54

Kulkarni, V. R., & Desai, V. (2016). ABC and PSO: A comparative analysis. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1–7). IEEE.

Leite da Silva, A. M., Rezende, L. S., Honório, L. M., & Manso, L. A. F. (2011). Performance comparison of metaheuristics to solve the multi-stage transmission expansion planning problem. *IET Generation, Transmission & Distribution*, *5*(3), 360–367. 10.1049/iet-gtd.2010.0497

Liapis, A., Yannakakis, G. N., & Togelius, J. (2013). Sentient sketchbook: Computer-assisted game level authoring. In *International Conference on Foundations of Digital Games*. Association for Computing Machinery. http://julian.togelius.com/Liapis2013Sentient.pdf

Lima, C. A. S.Jr, & Lapa, C. M. F. (2011). Comparison of computational performance of GA and PSO optimization techniques when designing similar systems typical PWR core case. *Annals of Nuclear Energy*, *38*(6), 1339–1346. 10.1016/j.anucene.2011.02.002

Liu, Y., & Tang, S. (2017). An application of artificial bee colony optimization to image edge detection. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC- FSKD)* (pp. 923–929). IEEE. 10.1109/FSKD.2017.8393400

Man, K.-F., Tang, K.-S., & Kwong, S. (1996). Genetic algorithms: Concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics*, *43*(5), 519–534. 10.1109/41.538609

Manning, T., Sleator, R. D., & Walsh, P. (2013). Naturally selecting solutions: The use of genetic algorithms in bioinformatics. *Bioengineered*, *4*(5), 266–278. 10.4161/bioe.2304123222169

Moghadam, A. B., & Rafsanjani, M. K. (2017). A genetic approach in procedural content generation for platformer games level creation. In *2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)* (pp. 141–146). IEEE. 10.1109/CSIEC.2017.7940160

Mora, C., Jardim, S., & Valente, J. (2021). Flora generation and evolution algorithm for virtual environments. In *16th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). IEEE. 10.23919/CISTI52073.2021.9476450

Mourato, F., dos Santos, M. P., & Birra, F. (2011). Automatic level generation for platform videogames using genetic algorithms. In ACE '11: *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology* (pp. 1–8). Association for Computing Machinery. 10.1145/2071423.2071433

Pal, A., Chan, F., Mahanty, B., & Tiwari, M. (2011). Aggregate procurement, production, and shipment planning decision problem for a three-echelon supply chain using swarm-based heuristics. *International Journal of Production Research*, *49*(10), 2873–2905. 10.1080/00207541003730847

Panda, M. (2018). Performance comparison of genetic algorithm, particle swarm optimization and simulated annealing applied to tsp. *International Journal of Applied Engineering Research: IJAER*, *13*(9), 6808–6816.

Paszkowicz, W. (2013). Genetic algorithms, a nature-inspired tool: A survey of applications in materials science and related fields: Part ii. *Materials and Manufacturing Processes*, *28*(7), 708–725. 10.1080/10426914.2012.746707

Pereira, L. T., & Toledo, C. F. (2017). Speeding up search-based algorithms for level generation in physics-based puzzle games. *International Journal of Artificial Intelligence Tools*, *26*(5), 1760019. 10.1142/S0218213017600193

Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, *63*(3), 337–370. 10.1007/BF02125403

Prasetya, H. A., & Maulidevi, N. U. (2016). Search-based procedural content generation for race tracks in video games. *International Journal on Electrical Engineering & Informatics*, *8*(4), 775–787. 10.15676/ijeei.2016.8.4.6

Reche-López, P., Ruiz-Reyes, N., Galán, S. G., & Jurado, F. (2009). Comparison of metaheuristic techniques to determine optimal placement of biomass power plants. *Energy Conversion and Management*, *50*(8), 2020–2028. 10.1016/j.enconman.2009.04.008

Sahu, C., Kumar, P. B., & Parhi, D. R. (2018). An intelligent path planning approach for humanoid robots using adaptive particle swarm optimization. *International Journal of Artificial Intelligence Tools*, *27*(5), 1850015. 10.1142/S021821301850015X

Schmidt, F., MacDonell, S., & Connor, A. M. (2018). Multi-objective reconstruction of software architecture. *International Journal of Software Engineering and Knowledge Engineering*, *28*(6), 869–892. 10.1142/S0218194018500262

Settles, M., Rodebaugh, B., & Soule, T. (2003). Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., & Miller, J. (Eds.), Lecture notes in computer science: *Vol. 2723*. *Genetic and Evolutionary Computation—GECCO 2003* (pp. 148–149). Springer., 10.1007/3-540-45105-6_17

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *IEEE Access : Practical Innovations, Open Solutions*, *10*, 10031–10061. 10.1109/ACCESS.2022.3142859

Skjærseth, E. H., Vinje, H., & Mengshoel, O. J. (2021). Novelty search for evolving interesting character mechanics for a two-player video game. In *GECCO '21: Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 321–322). Association for Computing Machinery. 10.1145/3449726.3459444

Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelbäck, J., & Yannakakis, G. N. (2010). Multiobjective exploration of the StarCraft map space. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games* (pp. 265–272). IEEE. 10.1109/ITW.2010.5593346

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172–186. 10.1109/TCIAIG.2011.2148116

Wang, L., Zhang, X., & Zhang, X. (2019). Antenna array design by artificial bee colony algorithm with similarity induced search method. *IEEE Transactions on Magnetics*, *55*(6), 1–4. 10.1109/TMAG.2019.2896921

Wang, Q., Spronck, P., & Tracht, R. (2003). An overview of genetic algorithms applied to control engineering problems. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)* (*Vol. 3*, pp. 1651–1656). IEEE. 10.1109/ICMLC.2003.1259761

Yannakakis, G. N., Togelius, J., Yannakakis, G. N., & Togelius, J. (2018). Generating content. In *Artificial intelligence and games* (pp. 151–202). Springer., 10.1007/978-3-319-63519-4_4

Youssef, H., Sait, S. M., & Adiche, H. (2001). Evolutionary algorithms, simulated annealing and Tabu search: A comparative study. *Engineering Applications of Artificial Intelligence*, *14*(2), 167–181. 10.1016/S0952-1976(00)00065-8

Yücel, F., & Sürer, E. (2020). Implementation of a generic framework on crowd simulation: A new environment to model crowd behavior and design video games. *Mugla Journal of Science and Technology*, *6*(2), 69–78. 10.22531/muglajsci.706841

Yusup, N., Zain, A. M., & Hashim, S. Z. M. (2012). Overview of PSO for optimizing process parameters of machining. *Procedia Engineering*, *29*, 914–923. 10.1016/j.proeng.2012.01.064

Zafar, A., Mujtaba, H., & Beg, M. O. (2020). Search-based procedural content generation for GVG-LG. *Applied Soft Computing*, *86*, 105909. 10.1016/j.asoc.2019.105909

Zhang, H., Zeng, X., & Duan, L. (2018). Procedural content generation for room maze. In *2018 3rd Joint International Information Technology,Mechanical and Electronic Engineering Conference (JIMEC 2018)* (pp. 265–269). Atlantis Press.

Zhang, L., Wang, S., Zhang, K., Zhang, X., Sun, Z., Zhang, H., Chipecane, M. T., & Yao, J. (2018). Cooperative artificial bee colony algorithm with multiple populations for interval multobjective optimization problems. *IEEE Transactions on Fuzzy Systems*, *27*(5), 1052–1065. 10.1109/TFUZZ.2018.2872125

Zhang, R. (2022). Automatic generation of real-time animation game learning levels based on artificial intelligence assistant. *Scientific Programming*, *2022*, 1557302. Advance online publication. 10.1155/2022/1557302

Zhang, Y., Zhang, G., & Huang, X. (2022). A survey of procedural content generation for games. In *2022 International Conference on Culture Oriented Science and Technology (CoST)* (pp. 186–190). IEEE. 10.1109/CoST57098.2022.00046

Zheng, Q., Feng, B.-W., Liu, Z.-Y., & Chang, H.-C. (2021). Application of improved particle swarm optimisation algorithm in hull form optimisation. *Journal of Marine Science and Engineering*, *9*(9), 955. 10.3390/jmse9090955