PROCEDURAL CONTENT GENERATION USING META HEURISTICS APPROACHES: A COMPARISON STUDY

A thesis submitted to

Auckland University of Technology

in fulfillment of the requirements for the degree of

Doctor of Philosophy (PhD)

Supervisors:

Senior Lecturer Parma Nand

Professor Roopak Sinha

May, 2024

By

Sana Alyaseri

School of Engineering, Computer and Mathematical Sciences

Intellectual Property Rights

Copyright in the text of this thesis rests with the Author. Copies (by any process), either in full or in extracts, may be made only in accordance with instructions given by the Author and lodged in the library, Auckland University of Technology. Details may be obtained from the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the Author.

The ownership of any intellectual property rights which may be described in this thesis is vested in the Auckland University of Technology, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

Further information on the conditions under which disclosures and exploitation may take place is available from the Librarian.

© Copyright 2024. Sana Alyaseri

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no materials that were previously published or written by another person. or material, which, to a substantial extent, has been accepted. for the qualification of any other degree or diploma of a universities and other institutions of higher learning.

Signature of Candidate

Acknowledgements

First, I extend my deepest appreciation to Associate Professor Andrew Connor for the immense support provided throughout my Ph.D. journey. Your guidance and encouragement as the primary supervisor in the initial and second phases of this research have been invaluable. I would like to express my sincere gratitude to Prof. Roopak Sinha for taking on the role of primary supervisor during the third part of my PhD. Your guidance, valuable time, feedback, and encouragement played pivotal roles in shaping the success of my research. Thank you. I am also grateful to my secondary supervisors, Jan Kruse, who was supported by PGR9, and senior lecturer, Parma Nand, who became my secondary supervisor in the third phase of the project and became primary in the submission period for their valuable input and feedback, which greatly contributed to the improvement of my work. I would like to extend my heartfelt thanks to my family (Alaa, Fatima, Hussein, Mohammed and Abdullah), especially my husband, Alaa Aljanaby. Your unwavering support, understanding, and belief in me have been instrumental to my pursuit of this dream. I am immensely grateful for your presence in every aspect of my life. Thank you. I would also like to acknowledge and express my appreciation to my colleagues at the AUT. Your companionship, insightful discussions, and shared coffee breaks provided me with moments of relaxation and support whenever I needed them. Once again, I extend my deepest gratitude to everyone mentioned above for their unwavering support, guidance, and beliefs throughout this journey.

Abstract

Procedural content generation (PCG) has emerged as a powerful approach for automating game content creation, offering significant benefits in terms of cost reduction and time efficiency, compared to traditional game design and development processes. The use of PCG algorithms enables the automatic generation of various game elements including terrain, maps, stories, dialogues, quests, and characters. Although genetic algorithms (GAs) have been widely employed in PCG, there is a growing interest in exploring alternative metaheuristic algorithms that can further enhance content generation capabilities.

In recent years, particle swarm optimization (PSO) and artificial bee colonies (ABC) have gained attention as effective metaheuristic algorithms capable of delivering high-quality solutions and efficient optimization in diverse problem domains. However, their application in PCG remains relatively limited, with most studies focusing on GAs. This study challenges the conventional 'one-size-fits-all' approach to PCG by assessing the effectiveness of PSO and ABC, specifically for race track and map generation tasks. The objective is to showcase their task-specific advantages over GAs, with the potential to enhance efficiency and content quality within these defined domains.

To achieve this, a comparative analysis was conducted among three metaheuristic algorithms, GA, ABC, and PSO. The objective is to assess the effectiveness and performance characteristics of these algorithms in generating game levels. Compre-

hensive experiments were conducted by applying GA, ABC, and PSO to generate diverse levels, such as race tracks and map layouts. Metrics, such as convergence speed and content quality, were employed to evaluate the generated game content. Convergence speed measures how quickly the algorithms reach optimal or near-optimal solutions, whereas content quality assesses the aesthetic appeal, playability, and overall suitability of the generated content for gameplay.

The findings of this study indicate that both ABC and PSO demonstrate advantages over traditional GA implementations in terms of race track generation. This highlights the potential benefits of integrating alternative metaheuristic algorithms into PCG workflow. By leveraging a diverse range of algorithms, PCG can not only improve content creation efficiency and effectiveness but also enhance the overall gaming experience.

Furthermore, This study underscores the importance of a "Task-Specific Considerations" approach when selecting algorithms for PCG tasks. This approach emphasizes a meticulous analysis of the inherent complexity of each task. Factors such as the desired level of content quality and the required solution speed are crucial considerations in the algorithm selection. The "Task-Specific Considerations" approach encourages exploration beyond traditional GAs and acknowledges the distinct advantages offered by alternative metaheuristic algorithms like ABC and PSO. Each algorithm possesses unique strengths and weaknesses in key areas such as exploration, exploitation, consistency, and solution quality. Therefore, the judicious selection of the most suitable algorithm based on the specific requirements of a PCG project is essential for its successful implementation.

Contents

List of Tables

List of Figures

Abbreviations

PCG Procedural Content Generation

SBPCG Search Based Procedural Content Generation

GA Genetic Algorithm

PSO Particle Swarm Optimization

ABC Artificial Bee Colony

SBPCG Search Based Procedural Content

SA Sameninling Annealing

ACO Ant Colony Optimization

MA Miscellaneous Algorithm

SI Swarm Intelligence

BA Bio Inspired Algorithm

TS Tabu Search

EA Evolutionary Algorithm

Publications

List of Published\Submitted Research Articles

Manuscript	Publication\Submission	Contribution
1	Alyaseri, S.; Sinha, R.; Nand, P., 2023, Systematic Literature Review of Meta-heuristic Algorithms and their Application in Procedural Content Generation (PCG) in the Context of Computer Games, submitted to Swarm and Evolutionary Computation Journal, Under review after revision (SWEVO-D-23-00382R1)	Sana alyaseri: 90%, Roopak Sinha: 6%,Parma Nand: 4%
2	Alyaseri, S.; Connor, A.; 2024, Comparative Analysis of Meta-heuristic Algorithms for Procedural Race Track Generation in Games, Submitted to International Journal of Applied Metaheuristic Computing (IJAMC), Accepted (050224-105008)	Sana Alayseri: 90%, Andy Connor: 10%
3	Alyaseri, S.; Connor, A.; Sinha, R.; 2024, Exploring Metaheuristic Algorithms for Enhanced Game Map Generation in Procedural Content Generation, submitted to applied intelligence journal, under review (APIN-D-24-03898)	Sana Alyasei: 90%, Andy Connor: 6%, Roopak Sinha: 4%
4	Alyaseri, S.; Nand, P.; Sinha, R.; 2024, A Context-Aware Approach Framework to Algorithm Selection for Search Based Procedural Content Generation, (Accepted in The IEEE International conference on Industry 4.0, Artificial Intelligence, and Communication Technology, Bali, Indonesia July 04-06, 2024	Sana Alyasei: 90%, Nand Parma: 6%, Roopak Sinha: 4%

We, the undersigned, hereby agree to the percentages of participation to the chapters identified above.

Signatures of the contributors

Name	Signature
Sana Alyaseri	
Parma Nand	
Roopak Sinha	
Andy Connor	

Chapter 1

Introduction

This chapter serves as the foundation for this thesis by providing crucial background information on PCG in game development. It explores the crucial role of metaheuristic approaches in PCG, highlighting their significant impact and contribution to the field. Moreover, it explains the rationale behind this research by detailing the existing challenges and motivations, emphasizing the importance of addressing these issues and generating new insights.

Subsequent to this introductory exploration, this chapter presents the research questions that guided the investigation. These questions outline the key areas of focus and central topics addressed in this thesis. Following this, the chapter delves into the research design, revealing the chosen metaheuristics and the specific optimization techniques used.

Finally, This chapter provides an overview of the thesis structure, provides a roadmap for the reader, and shows the flow and interconnectedness of the following chapters. This serves as a guide for orienting the reader to navigate through the development of this research.

1.1 Background

1.1.1 PCG and Game development

PCG refers to the algorithmic creation of game content that requires minimal or indirect user inputs. The terms “generation” and “procedural” suggest that we are working with computer programs or algorithms that produce something. A PCG method can be executed by a computer, possibly with human assistance, and will produce an output [1]. While the concept of “content” is central to PCG, encompassing a broad spectrum of game assets including levels, maps, rules, textures, narratives, items, quests, music, weaponry, vehicles, and characters [2]. Notably,

the game engine itself and nonplayer character (NPC) behavior (NPC AI) were excluded from this definition [1].

Although PCG research utilizes Artificial Intelligence (AI) techniques [3], it remains a distinct field in game development [4]. The core principle of PCG is to automatically create game content using algorithms rather than relying on manual creation. However, Algorithms often employ randomization to produce outputs with desired statistical properties. This method finds application in generating various forms of media, including images, landscapes, and particularly, game levels. PCG exploits algorithms and mathematical principles to automate game content generation, thereby enabling developers to move beyond manually crafted assets. Instead, developers establish parameters and rules that guide the dynamic creation of in-game elements [5], [6], [7].

An analogy can be drawn between PCG and the digital artist's brush. However, unlike the artist, PCG automatically explores a vast range of creative possibilities using algorithms and rules to paint unique and ever-evolving landscapes on virtual canvas. Imagine a digital paintbrush that dips into a vast number of possibilities and brings them to life through a game, story, or even music. This "paintbrush" can generate anything from sprawling landscapes and intricate dungeons to unique enemy encounters and engaging storylines. PCG manifests in diverse forms, each contributing uniquely to the overall gaming experience. It encompasses different types of content generation, including:

1. Procedural level generation: It is centered on the dynamic creation of varied game levels, providing novel landscapes and challenges for each playthrough. This approach aims to cultivate unpredictability and encourage the exploration of the gaming experience.[8], [9], [10].

2. Procedural Narrative Generation: Algorithms dynamically craft narratives and quests, providing players with unique and personalized storytelling experiences [11], [12].

3. Procedural asset generation: The automated creation of in-game assets such as characters, objects, or textures enhances the visual richness of the gaming environment [13].

4. Procedural Sound Generation: Algorithms dynamically generate soundscapes and audio elements by adding an auditory dimension to procedurally generated worlds [14].

Moreover, PCG provides multifaceted benefits to game developers and players by fostering innovation and efficiency.

First, it streamlines development pipelines by automating content creation, thereby enabling developers to concentrate on the core mechanics and narrative design. This efficiency translates to reduced development time and costs, contributing to a more sustainable game development industry [15]. Moreover, PCG empowers developers to break free from creative constraints, breed new genres, and reinterpret established tropes. Creative freedom enriches the gaming landscape by introducing novel captivating experiences.

Second, it enhances player engagement because the dynamic and unpredictable nature of procedurally generated content keeps players engaged and invested. The constant elements of surprise and discovery lead to longer playtimes and stronger emotional connections with the game [6].

1.1.2 PCG Approaches

The video game industry is increasingly turning to PCG to automate game content creation. This technology enhances player experience through diverse and dynamic gameplay by automatically generating various game elements. For example, games like “Spelunky” or “No Man’s Sky” utilize PCG to create unique level layouts for each playthrough, offering a fresh experience every time [16], [17]. Similarly, games like “Minecraft” leverage PCG to generate vast and diverse landscapes, encouraging players to explore and discover new locations and encounters [18].

Although a single, universally accepted taxonomy for PCG does not exist, researchers have frequently classified methodologies according to their key features. These consist of the type of content produced, degree of control granted to designers, and content representation [1].

Hendrikx et al. (2013) introduced a taxonomy that categorised PCG techniques according to the kind of content produced (see Figure 1.1). Six groups were identified using this classification system. This study focused on the Artificial Intelligence (AI) group within this taxonomy [19]. Three major AI approaches that are crucial to PCG have been identified in this field.

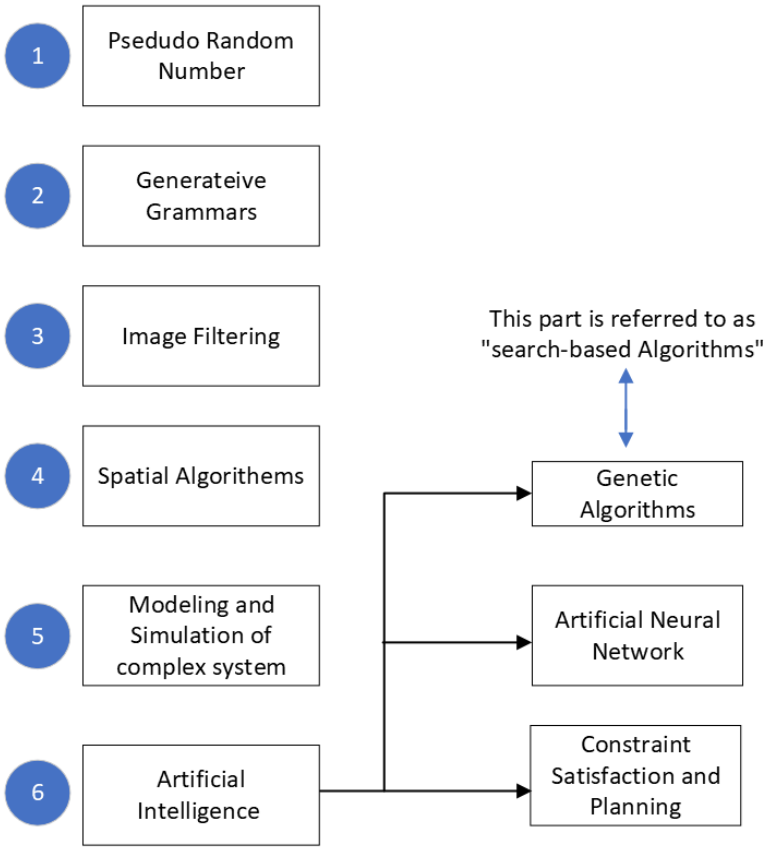


Figure 1.1: Artificial Intelligence Group in Hendrikx Taxonomy

The initial subgroup classified within AI approaches pertains to GA associated with a search-based optimization technique introduced by Goldberg in 1989 [20]. GAs draw inspiration from the principles of biological evolution to solve optimization problems. Potential solutions are encoded as strings (chromosomes) and their quality is assessed using a predefined fitness function. The algorithm iteratively evolves these solutions by applying the mutation and crossover operations. The mutation alters existing solutions, whereas crossover combines elements from different solutions to create new ones [20]. This process continues until a satisfactory solution is found or a predefined termination criterion is met, mimicking the natural selection process.

The next subgroup is Artificial Neural Networks (ANNs), introduced by Haykin in 1994 [21]. ANNs are computational models that are inspired by the structure and function of the human brain. They consist of interconnected processing units, called neurons, with weighted connections that influence the flow of information throughout the network [21]. By adjusting these weights based on the training data, the ANNs can learn the complex relationships between the inputs and outputs. This makes them well suited for tasks such as pattern recognition, classification, and data structuring in the context of PCG.

The Constraint Satisfaction and Planning (CSP), as outlined by Russell et al. in 1995 [22], represents the final subgroup categorized within the AI approaches of the Hendrikx Taxonomy [19]. CSP techniques focus on finding a sequence of actions that lead from an initial state to a desired end state while satisfying a set of predefined constraints, which are typically defined by an initial state, available actions, and a goal test that determines whether a particular state satisfies the desired outcome. Planning Domain Definition Language (PDDL) is a common way to address these problems. Various search algorithms such as forward and backward state-space searches have been employed to navigate through the possible solution space. Owing to the inherent complexity of planning problems (often classified as NP-hard), heuristics may play a crucial role in efficiently guiding the search for a solution.

However, the term “Search-based Approaches” refers to a broader category of techniques used in PCG, which includes GA as one of its subsets. These approaches are characterized by their method of systematically searching through a space of possible solutions to determine the most suitable or optimal solution [23], [24]. These approaches have developed from the traditional method of “generate and test” used in PCG. In the “generate and test” approach, potential solutions are created and

then evaluated to determine their suitability based on predefined criteria [23], [25].

Search-based approaches in PCG enhance the traditional "generate and test" method by incorporating advanced metaheuristic techniques such as GAs, simulated annealing, and PSO. These methods enable more efficient exploration of the solution space, leading to higher-quality solutions compared to simple "generate and test" approaches [1]. These approaches are based on two fundamental principles. First, they utilize a test function, often called a fitness, evaluation, or utility function, to assess selected content using numerical values. Second, they generate new candidate content with potentially better values based on the test function using algorithmic approaches. Although search-based approaches are commonly associated with population-based evolutionary methods, other algorithms can operate at a single search point [23], [1], [25].

The primary challenge in search-based PCG is evaluating the quality of game content items, which is accomplished primarily through the deployment of fitness functions. These functions assign a numerical score to each candidate piece of content, reflecting how well they meet the desired criteria. These functions fall into three key categories as outlined by Togelius et al. (2011) [23] See Figure 1.2.

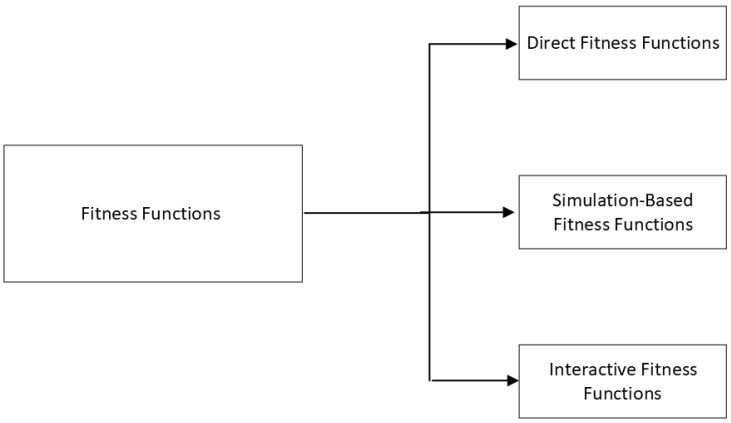


Figure 1.2: Fitness Functions Types

The first category within fitness functions encompasses Direct Fitness Functions. These functions directly assess the features of content based on predefined rules or metrics. For example, a fitness function might evaluate the balance at the game level by analyzing enemy placement and resource distribution. These elements can be given weights according to how crucial they are to reach a state of balance, and based on how effectively the level design complies with these predetermined criteria, the function then assigns a score. Our study is inspired by this category. We designed evaluation functions that directly assessed the specific characteristics of the content that we aimed to generate. These functions were meticulously tailored to align with the specific objectives of our study.

The second category of fitness functions comprises simulation-based fitness functions that employ gameplay simulation of the generated content to assess its quality. This approach provides a more dynamic evaluation than direct fitness functions. Factors, such as player behavior, level completion time, and AI performance, can be measured during the simulation to determine the fitness score.

Interactive Fitness Functions are the third type of fitness functions outlined by Togelius et al. (2011) [23]. These functions rely on human interactions to evaluate

content. Players may be asked to play through the generated content and provide feedback, which is then used to calculate the fitness score.

Although search-based methods are widely applicable to creating game content, their adaptability is limited. First, these techniques can be computationally costly because they must evaluate a large number of potential content items. Because most search-based algorithms lack runtime guarantees, it is often impossible to predict with certainty how long it takes to find an optimal solution. In addition, because these algorithms are stochastic, their convergence characteristics may not always be accurate. The evaluation function and algorithm representation must be carefully selected if these methods are effective [26]. The next section discusses specific types of algorithms associated with this subcategory of AI within the Hedrixne Taxonomy [19].

1.1.3 Metaheuristic Optimization Approaches

This subsection explains the metaheuristic optimization algorithms and briefly discusses the different types. Subsequently, the specific metaheuristics selected for inclusion in this study are enumerated.

1.1.3.1 Overview of Metaheuristics Approaches

The term “metaheuristic,” initially introduced by Glover in 1986 [27], combines two Greek words. “Heuristic” is derived from the verb “heuriskein”, meaning ’to find,’ while the prefix “meta” signifies ’beyond’ or ’at a higher level.’ Metaheuristic approaches draw inspiration from both natural and computational processes, and provide efficient solutions for intricate problems in which exact solutions pose challenges. These versatile algorithms find applications across diverse domains such as optimization, machine learning, and PCG. They are classified as nature-inspired approaches (NIAs) and replicate natural behaviors to solve optimization problems [28].

Metaheuristic optimization algorithms or metaheuristics constitute a broad class of optimization techniques that enable the discovery of optimal or near-optimal solutions for a wide array of problems. These algorithms guide the search process and explore the entire search space. The fundamental steps remained consistent across most metaheuristics. The initial stage involves generating an initial set of solutions (population) and evaluating the objective function for each solution. Subsequently, the algorithm performs an iterative search for the best solution, generating new solutions by modifying and combining the existing solutions. The implementation specifications were based on the mathematical model of each algorithm. Typically, new solutions replace their predecessors when the objective function value improves. The optimization algorithm concludes by reaching the specified stop condition [29].

In the PCG domain, metaheuristic algorithms play a pivotal role in automating the creation of engaging content for video games. A prime example is the application of Evolutionary Algorithms, with GAs as a notable instance. These algorithms mimic inherited behaviors, starting with a randomized population and evolving over several generations. GAs have demonstrated success in computer game development, particularly in enhancing Dungeon game levels through concept-based map generation [30], [31], [1].

Another approach within PCG is Swarm Intelligence (SI) techniques, which draw inspiration from animal behavior. Important examples include Ant Colony Optimization (ACO), ABC, and PSO. These techniques employ populations that simulate the behavior of ants, honeybees, and birds. The fundamental principles of self-organization and division of labor in SI algorithms closely mirror those observed in natural systems [32],[33]. Importantly, our research involves both ABC and PSO as representative examples in Swarm Intelligence.

Simulated Annealing (SA) is also considered a metaheuristic algorithm. This mirrors the metallurgical annealing process that begins with a random solution and iteratively explores the solution space. SA has found applications in PCG that strike a balance between exploration and exploitation, thereby contributing to the creation of dynamic gaming elements [34].

Examples in the PCG domain include ABC, ACO, PSO, Harmony Search (HS), and GA, which operate within a heuristic metaheuristic framework. Growing interest in metaheuristic algorithms for PCG has arisen from their distinct advantages over traditional approaches. These algorithms enable game developers to generate diverse and adaptable game content by leveraging computational power for innovative game designs. The selection of a specific metaheuristic depends on the characteristics of the PCG task, such as content type, desired exploration level, and game design constraints. Metaheuristic algorithms significantly contribute to pushing the boundaries of achievement in dynamic and evolving fields of game development. This thesis focuses on three key heuristic approaches that are considered efficient and successful: GA, PSO, and ABC. The following subsection provides an in-depth exploration of these approaches.

1.1.3.2 Description of Chosen Metaheuristics Approaches

This subsection presents a concise summary of the three algorithms: GAs, PSO, and artificial bee colony (ABC) used for search-based content generation in our study. A comprehensive detials and in depth description of these algorithms is provided in Chapters 5 and 7. See Figure 1.3 for a visual representation of these algorithms.

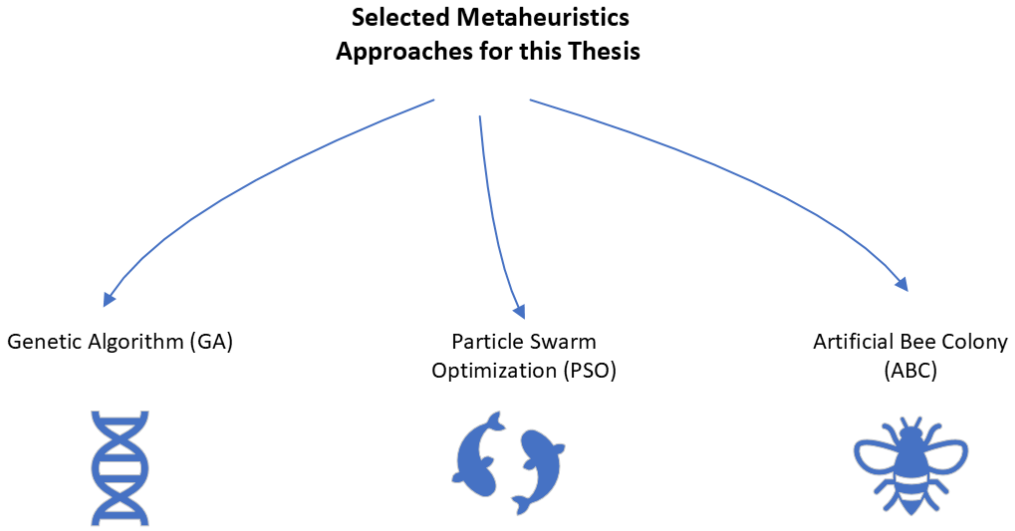


Figure 1.3: Selected Metaheuristics Approaches

1. The Genetic Algorithm (GA) [20] is a renowned evolutionary algorithm that draws inspiration from natural selection processes. It initiates with the random generation of an initial population, assesses the objective function for each individual, and results in a binary vector. Subsequently, the GA executes the crossover and mutation operations. Crossover involves creating new individuals by merging portions of the chromosomes from two distinct individuals, while mutation alters a randomly selected gene within an individual. The next generation comprises individuals with the best objective function values, and has found widespread application in PCG, particularly in domains such as level design for video games [35]. Its ability to explore a vast search space and discover optimal or near-optimal solutions makes it well suited for creating diverse and engaging game levels. Chapters 5 and 7 provide more detailed insights into the application of GA in race track and map generation tasks.

2. PSO [36] draws inspiration from the collective behavior of bird flocks in nature, representing a classic metaheuristic algorithm rooted in swarm intelli-

gence. Unlike traditional populations, PSO focuses on generating location and velocity vectors for individuals at the initial stage. During each iteration, PSO identifies the best-known position for each particle and the overall swarm by utilizing this information to calculate new particle velocity values. Consequently, the optimal state of the swarm serves as a solution to the optimization problem. In the context of PCG, PSO has demonstrated effectiveness in tasks such as race tracking and map generation in video games. Its ability to balance exploration and exploitation within a solution space makes it applicable to crafting diverse and playable game environments. Further insights into the application of PSO to PCG tasks can be found in Chapters 5 and 7.

3. ABC [37] is a metaheuristic algorithm inspired by the foraging behavior of honeybees. In ABC, the optimization process simulates the activities of employed bees, onlookers, and scouts to discover optimal solutions. Employed bees exploit food sources, onlookers assess their dances of employed bees to choose potential sources, and scouts explore new locations. This algorithm balances exploration and exploitation through its distinct roles. In PCG, ABC has shown its effectiveness in tasks such as race track and map generation for video games. By leveraging the principles of collective intelligence observed in natural bee colonies, ABC contributes to creating diverse gaming environments. Further details on ABC's application in PCG tasks can be found in Chapters 5 and 7.

1.2 Rationale and Significance of the Study

PCG has emerged as a powerful tool for game developers, offering significant time and cost benefits for content creation. Toelius [23] and Amato [15] demonstrated this potential, identifying PCG as a means to produce game content more efficiently by reducing human input. However, despite this promise, research by Connor et

al. [38] highlights the lack of understanding of how procedurally generated content impacts the overall gaming experience.

Similarly, a review of the approaches used in PCG has identified that few studies undertake a formal comparison of different approaches, and there is a bias towards using GAs when search-based approaches are adopted. Yet there is no clear answer why GAs are common as a PCG tool for games rather than other meta-heuristics approaches.

The popularity of GAs extends beyond PCG, serving as a well-established optimization approach in various domains [39], [40]. However, such research suggests that alternative algorithms may offer improved performance for specific tasks. For instance, studies in software engineering have demonstrated that PSO surpasses GAs in certain aspects [41].

An exhaustive study of algorithm performance across different domains is beyond the scope of this study; however, it is clear that no algorithm consistently performs better than any other algorithm. Some authors argue that the popularity of evolutionary approaches stems from a history of success rather than a conscious choice based on the characteristics of the problem at hand [42].

Although PCG has proven effective, there is room for improvement. Researchers have proposed incorporating a novel learning method that allows algorithms to explore and learn independently. This integration aims to enhance PCG further by combining established techniques that improve how algorithms find solutions [1] [43]. This innovative approach is expected to significantly reduce the time required to generate levels, particularly when a large number of valid levels are required.

Finally, Scheibenpflug (2016) [44] introduced an evolutionary paradigm for 2D map generation in computer games that incorporates user preferences and uses novelty search principles to produce a variety of diverse solutions. Additionally, it has been noted that in many cases, algorithms often use problem-specific information to enhance their performance [45], and it is evident that the way a problem is represented can impact performance [46].

Therefore, it is necessary to determine whether the popularity of GA in PCG is justified or whether studying different approaches could be open areas for researchers in the field of metaheuristics to step on and use these approaches in developing automated content generation to leverage any advantage offered by alternative algorithms. The advantages of using alternative algorithms in PCG tasks include the following.

- Diversity and Performance: Alternative algorithms can provide diversity in level structure, which is essential for preventing player boredom and promoting varied gameplay in multiplayer games [47], [48].
- Efficiency and Effectiveness: Alternative algorithms, such as Quality-Diversity (QD) algorithms, can lead to good performance in generating diverse content efficiently, which is essential for creating a broad variety of content in an efficient manner[47].
- Enhanced Productivity: Increased automation with alternative algorithms, particularly through artificial intelligence, can enhance productivity and effectiveness in generating game content [49].

These advantages highlight the potential of alternative algorithms to improve the diversity, performance, and efficiency of PCG in various applications.

1.3 Problem Definition and Motivation

The video game industry has undergone a significant paradigm shift with the integration of metaheuristic algorithms into procedural content generation (PCG). These algorithms, inspired by natural phenomena and computational models, exhibit remarkable flexibility and adaptability. Their ability to navigate complex solution spaces and generate diverse content makes them particularly well-suited to the dynamic landscape of video game development.

Metaheuristic algorithms in PCG present substantial potential by enabling creative solutions that traditional content generation techniques may struggle to achieve. Unlike deterministic algorithms that follow predefined paths, metaheuristics, such as GA, PSO, and ABC leverage randomness and exploration to discover novel and optimized content configurations. This capability enhances the originality and diversity of generated content, ensuring engaging and immersive gaming experiences.

The significance of metaheuristic algorithms in content generation is particularly evident when addressing the need for diversity and uniqueness to maintain player engagement. These algorithms introduce variability and unpredictability, contributing to dynamic and evolving game worlds that sustain player interest over extended periods.

Moreover, metaheuristic algorithms optimize content generation processes by improving efficiency and reducing development time. Their dynamic nature allows for automated adaptation of generated content to meet predefined criteria and constraints, resulting in refined and polished game elements. This adaptability is crucial in modern game development, where procedural techniques are increasingly employed to generate levels, characters, textures, and other game components.

Despite the broad range of available metaheuristic algorithms, Genetic Algorithms have dominated the PCG landscape due to their ability to mimic natural

selection and evolution. Their iterative approach to evolving content over successive generations has made them highly effective for various content types, including levels, maps, and textures [50]. However, while GAs have demonstrated success, their widespread use prompts an exploration of alternative metaheuristic approaches. PSO and ABC, for example, offer different mechanisms for search and optimization, potentially addressing limitations of GAs and expanding the scope of PCG methodologies.

This study bridges a critical knowledge gap in the existing literature by conducting a novel comparative analysis of the performance of metaheuristic algorithms in PCG for game development. While prior research has explored the utilization of metaheuristics in PCG, there remains a lack of comprehensive comparisons evaluating their effectiveness in terms of quality and convergence metrics, particularly in generating race track levels and map layouts. This study directly addresses this need by comparing the performance of GA, ABC, and PSO for these specific content types.

Beyond addressing this knowledge gap, the study contributes significantly to the field of PCG by:

- Providing a comprehensive comparative study that directly evaluates the performance of three metaheuristic algorithms in race track and map layout generation. This research offers valuable insights into the strengths, limitations, and applicability of these approaches for game content generation.

- Assisting game developers in making informed decisions about algorithm selection based on performance characteristics. By understanding the strengths and weaknesses of each algorithm, developers can enhance efficiency and effectiveness in content generation, ultimately leading to richer and more immersive gaming experiences.

- Encouraging exploration of alternative metaheuristic approaches beyond GAs.

Expanding the range of algorithms used in PCG can inspire innovative game design strategies and unlock new creative possibilities.

- Motivating further research and experimentation with different metaheuristic algorithms. By broadening the scope of PCG techniques, this study paves the way for more dynamic, engaging, and adaptive game content.

The incorporation of metaheuristic algorithms into PCG presents a powerful framework for addressing content generation challenges, optimizing creation processes, and introducing viable alternatives to established methods. While GAs have played a prominent role in PCG, exploring other metaheuristics holds immense potential for enriching the content generation landscape and advancing the field of game development.

1.4 Research Questions

The aim of this study was to investigate the efficacy of different metaheuristic algorithms in the domain of PCG, with a specific focus on race track and map generation tasks. This section outlines several core research inquiries and explores specific aspects of the comparative analysis, as follows:

- RQ1: What are the most common and effective meta-heuristic algorithms used for PCG in games and how have these algorithms been applied across various domains within PCG?
- RQ2: What are the main challenges and limitations of using meta-heuristic algorithms for PCG and how can these be addressed in future research?
- RQ3: How do GA, ABC, and PSO compare in terms of solution quality and convergence speed when applied to the task of race track generation in PCG?

- RQ4: In the context of map generation for PCG tasks, how are GA, ABC, and PSO compared in terms of solution quality and convergence speed?
- RQ5: How stable are the solution quality and convergence speed of GA, ABC, and PSO when applied to race track and map applications?
- RQ6: How applicable are the findings of this comparative study to a broader range of PCG tasks beyond those specifically examined, such as race track and map generation?

The research questions directed the methodology of this study, which involved designing two comparative PCG tasks for the chosen metaheuristic algorithms, collecting data through experimentation with these algorithms and tasks, and conducting analyses to provide insights into the comparative effectiveness, advantages, and applicability of metaheuristic algorithms across a broader field of PCG.

1.5 The Research Design

PCG, which includes characters, levels, and storylines, is becoming increasingly important in video games. Traditionally, GAs have been the main optimization methods in this field. However, recent studies have indicated that other metaheuristic search algorithms may provide notable benefits in terms of effectiveness and speed. This study attempted to illuminate this potential by thoroughly assessing and analysing the efficacy of different metaheuristics in PCG challenges.

Although GAs remain prevalent in PCG because of their established effectiveness, growing research has explored the potential of alternative algorithms. This study contributes to this investigation by conducting a novel comparative analysis of three prominent metaheuristics: PSO, ABC, and GAs. This comparative approach allowed for a nuanced understanding of the relative strengths and weaknesses of each

algorithm in the context of PCG tasks.

The investigation primarily focused on a quantitative analysis by employing key metrics to assess the performance of the algorithms. These metrics include:

- **Fitness function values:** This indicator gauges the quality of the solutions generated by each algorithm.
- **Convergence speed:** This metric measures the efficiency of an algorithm in finding optimal solutions, focusing on how quickly it converges to a desirable outcome.

The stochastic nature of the selected algorithms required the generation of multiple solutions for each PCG task. Descriptive statistics [51] were employed to analyze the performance differences between the algorithms. This approach ensures a thorough exploration of the generated data, while providing valuable insights into the relative strengths and weaknesses of each algorithm.

To facilitate a fair comparison, the experimental design was consistent across all variables except for the algorithm. This meticulous control ensures that any observed differences in the performance can be directly attributed to the specific characteristics of each algorithm. This rigorous approach allows for a clear and unbiased evaluation of the effectiveness of the algorithms in generating high-quality game content.

Although playability is a crucial aspect of game design, its deliberate exclusion from the scope of primary evaluation serves a specific purpose. The fitness function used for individual-level evaluation inherently incorporates elements related to playability. Any modifications or improvements to this function are consistently reflected across all algorithms, thereby minimizing any potential impact on individual

performance comparisons. This focus on core algorithmic performance allows for a deeper understanding of the specific strengths and weaknesses of each approach in the context of PCG tasks.

This study used quantitative data analysis through a series of computational experiments to answer research questions (RQ1, RQ2, RQ3, RQ4, RQ5, and RQ6) regarding the application of metaheuristic algorithms in PCG. Figure 1.4 illustrates this three stages of the research design. The first stage involves a thorough review of existing research in this area. The next stage focused on the chosen algorithms. Specific tasks were designed for these algorithms. Then, the algorithms generate numerical results based on these tasks. Finally, we carefully analyze these results to compare them and see how well they answer the research questions such as effectiveness and efficiency. This combined approach ensures that we fully explore the research goals, ultimately leading to a better understanding of the different metaheuristics used in the PCG.

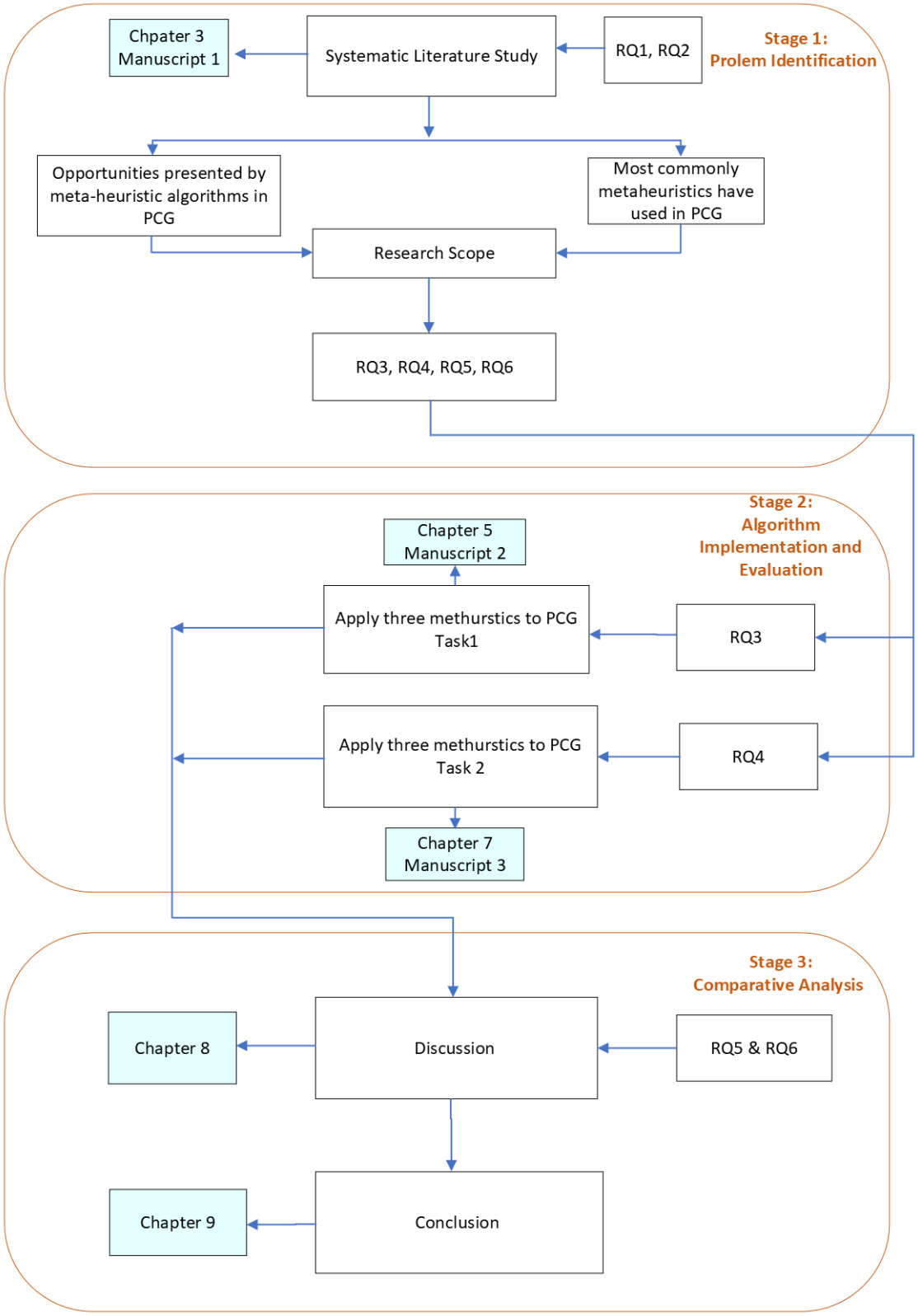


Figure 1.4: Research Design

1.5.1 Stage One: Systematic Literature Review

The first stage employed a quantitative and qualitative approach [52] through a systematic literature review. This review specifically addresses the research questions RQ1 and RQ2. By analyzing the existing research published between 2007 and 2021, this review focuses on investigating the approaches employed in metaheuristic methodologies for PCG tasks. This critical analysis served two primary purposes.

1. **Foundations for Understanding:** This provides a comprehensive understanding of the current trends in metaheuristic applications within PCG. This allowed the study to identify prevailing approaches and potential research gaps in this field.

2. **Knowledge Base for Further Exploration:** The review is equipped with a solid knowledge base to effectively navigate the subsequent stages. By building this foundation, this study ensures a more informed and targeted approach in later stages.

Chapter 3 provides a detailed exploration of the review methodology and its findings. This chapter meticulously examines the review process, offering valuable insights into the current landscape of PCG and metaheuristic methods.

1.5.2 Stage Two: Algorithm Implementation and Evaluation

Building on the foundation laid out in the literature review, the second stage of the research involved empirically exploring the selected metaheuristic approaches. This stage assessed their effectiveness and efficiency in generating game content through two distinct PCG applications, specifically focusing on the generation of race tracks and map maps. Given the stochastic nature of the algorithms, multiple runs were conducted for each algorithm and the PCG application. This repetition ensures a more robust understanding of their performance and accounts for potential

variations in the outcomes. Therefore, to evaluate each run comprehensively, the following metrics were used.

- **Best Solution:** This represents the solution with the best fitness function value, signifying its perceived quality amongst the generated levels for each algorithm.
- **Worst Solution:** This metric represent the worst fitness value
- **Number of Iteration:** This metric reflects the algorithm's efficiency, indicating the computational effort (number of iterations) required to reach the best solution.
- **Standard Deviation:** An indicator of how much a random variable is predicted to vary from its mean.

The evaluation process consists of two steps. First, an inspection step observes the distribution of the results for each approach and PCG application, evaluating the best, worst, and standard deviations of the fitness function values over several runs. This approach, which was used in earlier studies [53], offers insights into the effectively (quality of solutions) and efficiency (reliability in producing quality solutions) of algorithms. This assessment was conducted during the statistical analysis phase. The raw data were subjected to a statistical significance assessment. This approach provides strong evidence to support the conclusions by carefully evaluating whether the observed performance differences between algorithms are statistically significant.

Chapters 5 and 7 present detailed insights into the specific implementation, evaluation methods, and findings of this stage.

1.5.3 Stage Three: Comparative Analysis

Utilizing a quantitative methodology, the research's last stage explored the performance comparisons that began in Stage 2. This stage seeks to obtain a further understanding of and responses to RQ5 and RQ6 by examining the data gathered from the application of the selected metaheuristic approaches to PCG tasks. It aims to discern differences when applying these metaheuristics to PCG tasks. It investigates whether GA outperform other approaches in terms of convergence, and explores the performance of alternative metaheuristic algorithms across both tasks.

This structured and comprehensive approach, which combines systematic literature review, in-depth empirical exploration, and accurate quantitative analysis, ensures a holistic research journey. This journey provides valuable insights into the field of PCG and a method for advancements in this area.

1.6 Thesis Structure

The thesis follows the "Thesis in Manuscript" format, structured as outlined below:

1. Introduction Chapters (2, 4, and 6): These chapters provide an overview of manuscripts 1, 2, and 3 that follow. Each chapter corresponds to a specific phase of the methodology and presents its results, along with a literature review.

2. Chapter 3: Systematic Literature Review: This chapter conducts a systematic literature review on PCG in computer gaming. It assessed the use of metaheuristic approaches, explored the applications of different algorithms, and identified existing research gaps, thereby addressing RQ1 and RQ2.

3. Chapters 5 and 7:

(a) Chapter 5: Focuses on comparing three metaheuristic approaches applied to race track games. The goal was to evaluate the effectiveness of GA, ABC, and PSO in producing game content. This chapter addresses RQ3.

(b) Chapter 7: Delves into the results of the second phase of the project, dedicated to evaluating meta-heuristic algorithms in map layout generation for PCG. It compares GA, PSO, and ABC and analyzes their effectiveness and efficiency in generating game levels. This chapter addresses RQ4.

4. Chapter 8: Discussion: Discusses the overall research, highlighting major findings and limitations. Focuses on the third stage of the project and addresses RQ5 and RQ6.

5. Chapter 9: Prospects, Limitations and Conclusions: Provides concluding remarks and outlines potential directions for future research.

Each manuscript covers its respective research stages in detail, including background, methodology, results, and conclusions. This thesis aims to comprehensively evaluate metaheuristic algorithms in PCG with a focus on both race track and map layout generation tasks.

Chapter 2

Prelude Manuscript 1

This systematic literature review assessed the use of metaheuristic approaches in PCG by conducting a comprehensive literature review on computer gaming. We explored the applications of different metaheuristic algorithms and the most commonly used techniques. The study selection process is illustrated in Figure 2.1. Our findings demonstrate the popularity of GA in PCG and highlight the critical role played by algorithm parameters, problem complexity, and objective functions in determining their performance. Moreover, we highlight the opportunities presented by metaheuristic algorithms in PCG, such as improving game design, enhancing player experience, and reducing development time and costs.

Furthermore, Manuscript 1 highlights the challenges associated with using these algorithms, including the need for better evaluation metrics and balancing the trade-off between efficiency and effectiveness. We also suggest future research directions including developing new algorithms, exploring hybrid approaches, and addressing the ethical and social implications of PCG. Ultimately, manuscript 1 addresses theses research questions:

``What are the most common and effective metaheuristic algorithms used for PCG in games and how have these algorithms been applied across various domains within PCG?"

and

``What are the main challenges and limitations of using metaheuristic algorithms for PCG and how can these be addressed in future research?".

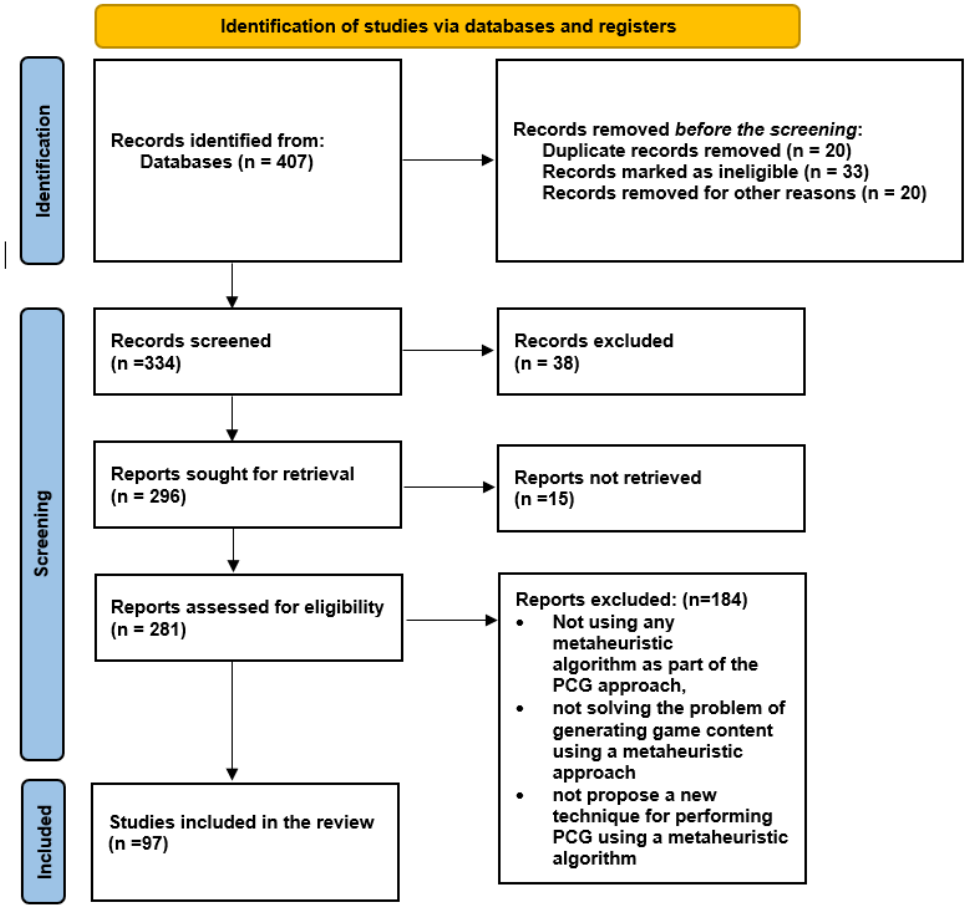


Figure 2.1: PRISMA Flow Diagram

Chapter 3

Systematic Literature Review of Meta-heuristic Algorithms and their Application in Procedural Content Generation (PCG) in the Context of Computer Games

3.1 Introduction

Digital games have become increasingly popular and complex. As a result, all forms of game content are in high demand and are becoming increasingly time consuming to develop. The complexity of computer games has steadily increased. Moreover, despite its high level, escalating expenses are linked to the development of manual video game content. The primary objective of procedural content generation (PCG) is to autonomously generate game content using an algorithmic approach. This strategic method significantly mitigates expenses associated with game design and development [5].

PCG algorithms can be applied to all types of gameplay, including terrain, maps, stories, dialogues, quests, characters, and others. Despite the limitations of conventional PCG algorithms in generating only one type of content, researchers have explored the use of metaheuristic algorithms as an alternative approach to game content generation [23]. Meta-heuristic algorithms are computational intelligence techniques used in challenging optimization problems and can generate multiple types of game content. These are a group of algorithms that draw inspiration from natural phenomena. These algorithms are commonly referred to as nature-inspired approaches (NIAs) because they attempt to replicate the behavior of natural systems to solve complex optimization problems [54].

Meta-heuristic algorithms include artificial bee colony (ABC), ant colony optimization (ACO), particle swarm optimization (PSO), harmony search (HS), and genetic algorithms (GA). Their techniques were developed using strategies specified in a meta-heuristic framework [55].

This article presents a comprehensive review of the different types of meta-

heuristic algorithms used in procedural content generation (PCG). It examines their strengths and limitations, and assesses their suitability for various PCG tasks. It explores the applications of meta-heuristic algorithms in the areas of PCG, such as level design, game mechanics, and character generation. In our view, this study has the potential to make significant contributions to the field of PCG and meta-heuristic algorithms by providing valuable insights and guidelines for their efficient application in PCG tasks. The contributions of this paper are as follows:

1. It presents various ways of using meta-heuristic algorithms in PCG, highlighting their potential benefits in enhancing game design and user experience.

2. It identifies the challenges associated with the application of meta-heuristic algorithms in PCG, such as issues related to algorithms, such as fitness function quality and the search mechanism.

3. It helps researchers identify potential research gaps and develop strategies to address these challenges.

The rest of this article is structured as follows. Section 3.2 provides a comprehensive background on the subject matter. Section 3.3 outlines the research methodology used to perform the systematic literature review. Section 3.4 presents the research findings and results. Finally, Section 3.5 concludes the article by summarizing the significant findings and highlighting the contributions of this study to the field of PCG and metaheuristic algorithms.

3.2 Background

The following sections provide essential background information to assist the reader in understanding the context, procedures, and metaheuristic algorithms discussed in this work.

3.2.1 Procedural Content Generation (PCG)

Procedural content generation (PCG) has been an integral part of game development since the advent of home computers. Notable early examples include Beneath Apple Manor from 1978 [56] and Akalabeth released in 1980 [15]. Since then, various commercial games have successfully utilised PCG methods. These games have their content algorithmically developed, either during the design process or, in some cases, during runtime, rather than being designed by a human. The definition of PCG has been discussed to clarify its meaning and highlight its limitations. Numerous definitions of PCG exist in literature. In the context of computer games, this involves automated or computer-assisted creation of in-game elements such as levels, landscapes, items, rules, and quests. Top game developers have been widely endorsed as a reliable method for enhancing gameplay experience [1].

PCG has been applied to the development of a wide range of game content, such as levels, adventures, characters, weapons, and histories [57]. Such content can be generated through many PCG approaches, some of which are based on methods from artificial intelligence (AI) and computational intelligence (CI), such as the use of evolutionary computation and constraint satisfaction [13]. PCG is a powerful tool for creating all types of game materials, such as environments, levels, and assets. Designers may create materials that look natural while adding enough variance to keep them from becoming repetitive or predictable using PCG [58]. To organize and understand the different PCG methods used in game development, Togelius et al. proposed a classification system [23].

This classification system categorizes PCG methods based on their scope or the types of problems they are best suited to solve. According to this system, the PCG methods can be categorized as follows:

- Online and Offline: Contents can be generated online while the player is actively playing the game, offering endless variation and replayability. Offline generation refers to the generation of content during the game-development stage.

- Necessary and Optional: Some generated content is necessary and forms a crucial part of the game rules, essential for player progress. Other content may be optional, allowing players to progress in the game without them, depending on the game design.

- Random Seeds and Parameter Vectors: Different mechanisms can be employed to control the generation of contents. This can include the use of a random seed in a random number generator for control, or using a vector of parameters representing the features or properties of the contents to be generated.

- Stochastic and Deterministic Generation: While deterministic PCG systems generate the same contents given the same inputs and parameters, stochastic PCG systems do not guarantee the same contents with identical inputs.

- Constrictive versus Generate and Test: Some PCG algorithms test the generated contents against specific criteria in a loop until matching contents are generated, combining generation and testing mechanisms. On the other hand, constructive PCG systems aim to generate optimal content from the start based on a set of rules and guidelines.

A taxonomy of PCG approaches has been developed by Zhang et al. [59] which describes the common methods used to generate game content for different game layers. This taxonomy was structured into three groups, as shown in 3.1. However, the inclusion of meta-heuristic approaches in the study has been omitted, possibly because of their categorization as a subset of search-based methods given the observed similarities between the techniques employed in search-based methods and

meta-heuristics.

There is growing interest in search-based approaches to PCG, which has become a key topic in procedural content generation. Many commercial games, such as Super Mario Bros, Pac-Man-Like, Dwarf Fortress, Star craft, and GVGAI, have been well established using these approaches.

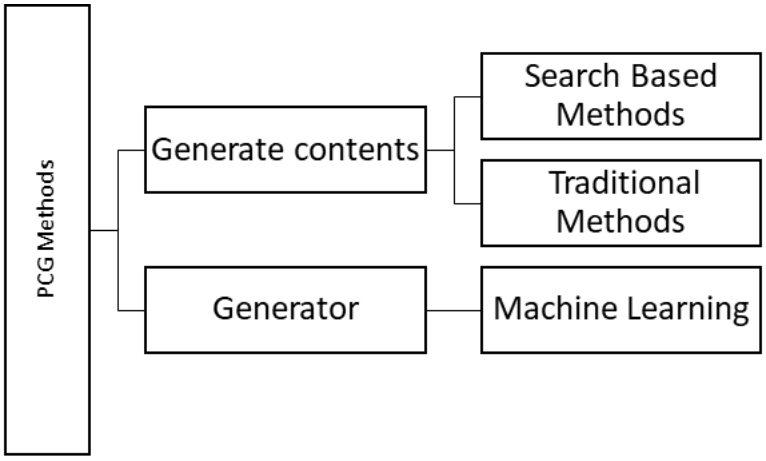


Figure 3.1: PCG Methods Taxonomy

3.2.2 Existing Meta-heuristics Algorithms

The field of meta-heuristics has significantly expanded over the last two decades in response to practical optimization issues. Meta-heuristics are effective in situations where traditional optimization techniques fail to produce the expected results. These techniques can generate high-quality solutions for complex optimization problems, including those classified as non-deterministic polynomial time-hard issues. Various fields such as finance, planning, scheduling, and engineering design have successfully utilized meta-heuristics [60]. Meta-heuristics belong to the domain of algorithms that draw inspiration from natural phenomena [61] and are sometimes referred to as NIAs [62].

The primary goal of implementing meta-heuristic algorithms is to identify the best solution among all the potential solutions to an optimization problem. To achieve this, meta-heuristic algorithms assess and perform several operations on possible solutions, iteratively generating new and improved solutions. A fundamental concept underlying meta-heuristics is the representation or encoding of a solution, which can be stored in computer memory and modified using various operators specific to the algorithm. This process forms the foundation of meta-heuristic operations, enabling efficient exploration and evaluation of potential solutions for complex optimization problems. By manipulating the solution representation through a series of operators, meta-heuristics iteratively improve candidate solutions until an optimal or near-optimal solution is obtained [63].

Meta-heuristic algorithms have introduced a variety of approaches. Each method may be more suitable for certain types of problems, and all the algorithms have limitations. Sharma and Tripathi have classified these approaches in 2022 based on the area they belong to which the research going to take as a point to start [61]. A classification categorizes algorithms inspired by nature into five groups: evolutionary algorithms (EA), physical algorithms (PA), swarm intelligence (SI), bio-inspired algorithms (BA), and miscellaneous algorithms (MA), as shown in Figure 3.2. Classification depends on focus, emphasis, and perspective. However, a brief overview of diverse group approaches is presented.

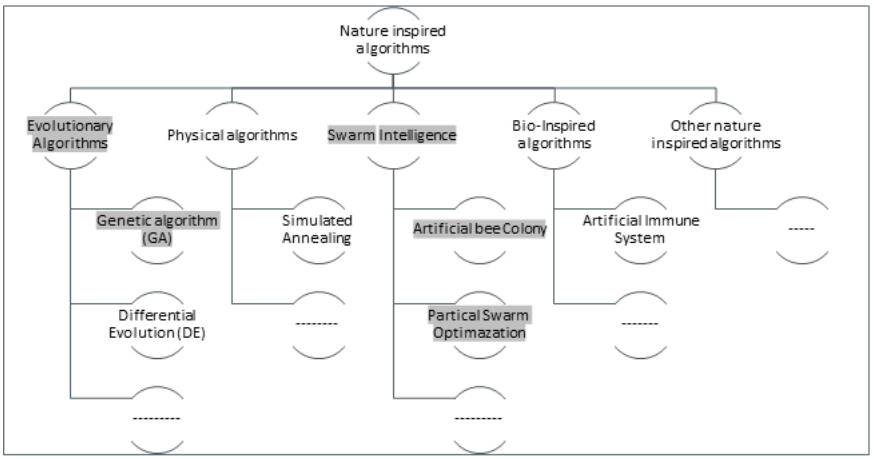


Figure 3.2: The classification of nature-inspired algorithms

3.2.2.1 Evolutionary algorithms

Evolutionary algorithms are inspired by nature and model the passing of hereditary traits from one generation to the next, like the concept of ”survival of the fittest” in Darwinian theory. These methods begin with a randomly initialized population and evolve the population over multiple generations. The most common example of evolutionary algorithms is genetic algorithms (GAs), which simulate natural selection and are widely used in engineering and science. GAs use terminology imported from biology, and the basic operations common to most GAs include fitness functions, populations of chromosomes, selection processes, crossover procedures, mutation actions, parents, and offspring. GAs iteratively improve candidate solutions based on their fitness until optimal or near-optimal solutions are found. GAs have been successfully applied to various problems in engineering, science, design, manufacturing, energy systems, and other domains, such as solving the Travelling salesman problem and optimizing thermoelectric modules (TEM) [64].

3.2.2.2 Physical algorithms

Physical algorithms are inspired by physical phenomena and incorporate ideas from fields such as chemistry, physics, music, and sports. Examples of physical-inspired meta-heuristics include simulated annealing, black hole algorithm, tabu search, cultural algorithm, harmony search, and teaching-learning-based optimization. These algorithms are based on principles derived from physical laws, chemical processes, and socioeconomic or demographic aspects. Simulated annealing (SA) and tabu search (TS) have gained significant prominence and have been successfully applied in industrial applications [65]. TS uses memory to explore previous solutions, remember the current best solution, and guide the search direction. Its adaptability and exploration capabilities enable it to find efficient solutions and new search spaces[66].

3.2.2.3 Swarm Intelligence Approaches (SI)

Swarm intelligence (SI) approaches involve a population of workers, particles, or agents working together to find optimal or near-optimal solutions. Examples of SI algorithms include artificial bee colony (ABC), ant colony optimization (ACO), and particle swarm optimization (PSO), in which populations simulate honeybees, ants, and birds, respectively. SI algorithms exhibit swarm intelligent behaviour based on self-organization and division of labour principles.

For instance, ABC has been effective in resolving various engineering and scientific problems, such as improving edge detection in digital image processing and solving antenna array design problems [37], [67], [68]. PSO, inspired by bird flocking and fish schooling behaviours, has also shown good performance in many applications, including solving non-oriented two-bin packing problems and determining moisture diffusion coefficients [69], [70], [71].

3.2.2.4 Bio-Inspired Algorithms (BA)

Bio-inspired algorithms mimic the behaviour of biological systems such as the artificial immune system or jellyfish hunting food. These algorithms aim to achieve high efficiency, even if an ideal outcome is not always attained. An example of a bio-inspired meta-heuristic algorithm is the jellyfish search optimizer (JSO), which takes inspiration from jellyfish hunting in the ocean [72].

3.2.2.5 Miscellaneous Algorithms (MA)

Miscellaneous algorithms encompass a range of approaches that are influenced by real-world applications. These algorithms involve making observations, designing mathematical modules, creating pseudocodes, and testing [61].

3.3 Research Methodology

This study utilized a systematic approach based on the methodology outlined by Kitchenham and Charters [73]. The methodology consists of a multiphase process that includes planning, searching, and reporting to comprehensively identify, classify, and analyse the existing literature within a specific area. Furthermore, this approach accounts for the contributions made to various categories within the area, providing a comprehensive analysis of the literature. The objective is to evaluate the use of various meta-heuristic approaches in creating gameplay content and to discuss the challenges in implementing these algorithms for PCG. Figure 3.3 illustrates the detailed steps in the structured methodology.

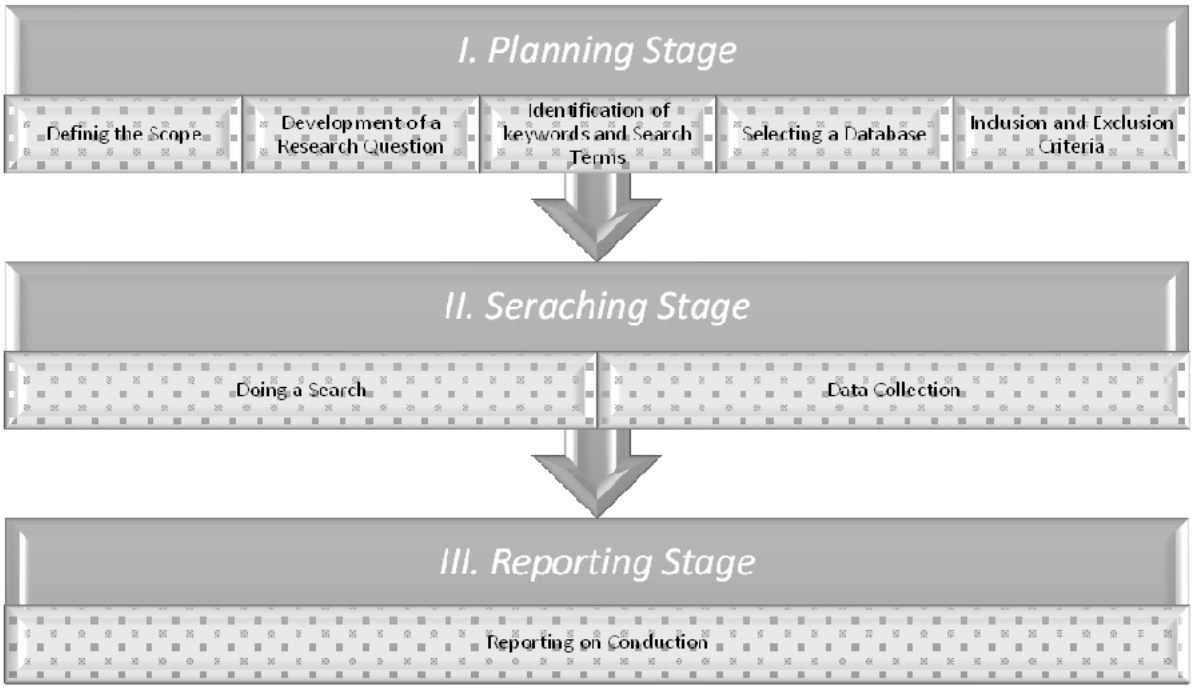


Figure 3.3: The research methodology

1. *Planning Stage*

   The planning stage of our methodology is crucial and involves a series of sequential steps to provide a well-defined framework for our investigation. This involves five fundamental components: defining the scope, developing a research question, identifying keywords and search terms, selecting the database, and determining the inclusion and exclusion criteria.

   (a) **Defining the Scope**

   Defining the scope of this study is a critical aspect of the planning process. This scope is described by the intersection of three primary domains: search-based procedural content generation methods, meta-heuristic approaches with a specific emphasis on algorithms inspired by natural processes, and the realm of computer games, Figure 3.4.

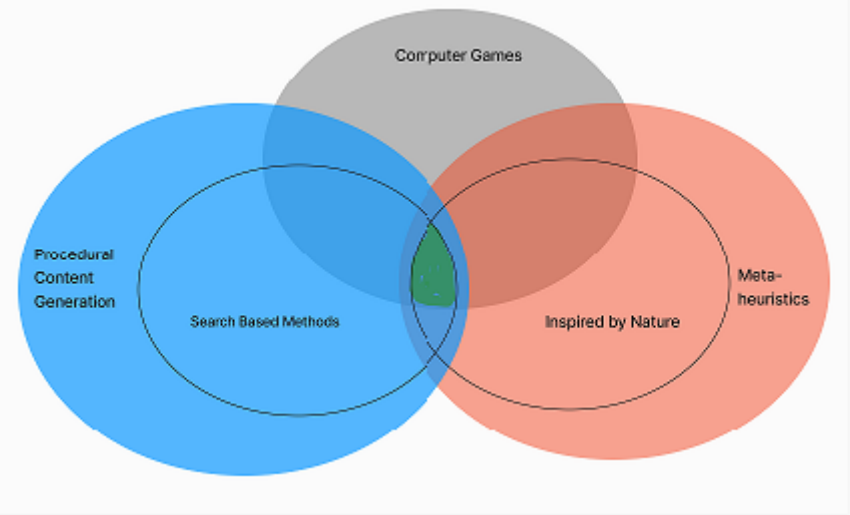


Figure 3.4: The scope of study

(b) **Developing the Research Questions**

Research question development, an essential step in directing research, is the focus of the second part of the planning stage. The scope leads to the following research questions:

RQ1:What are the most common and effective meta-heuristic algorithms used for procedural content generation (PCG) in games and how have these algorithms been applied across various domains within PCG?

This research question aims to identify and explore diverse meta-heuristic algorithms applied to procedural content generation (PCG). It provides a comprehensive overview of the algorithms used to generate game content using computational techniques. Additionally, it examines the frequent utilization of these metaheuristic algorithms in PCG and their applications across various domains within the field.

RQ2: What are the main challenges and limitations of using meta-heuristic algorithms for PCG and how can these be addressed in future research?

This research question specifically addresses the challenges and limitations that arise when meta-heuristic algorithms are employed in procedu-

ral content generation. It aims to identify the main obstacles and suggests potential solutions or opportunities for future research to overcome these challenges and enhance the effectiveness of meta-heuristic-based PCG methods.

(c) **Identifying the Keywords and Search Terms**

In the third phase of our planning process, we focused on identifying the most relevant keywords and search terms to capture appropriate studies effectively. To achieve this, we selected a range of keywords, including but not limited to "procedural content generation," "games," "computer games," "video games," "meta-heuristics," "inspired by nature," "nature-inspired," "evolutionary," "physical algorithms," "swarm intelligence," and "bio-inspired." These keywords served as the foundation for constructing our search terms, which were carefully formulated by combining them using logical operators such as "OR," "NOT," and "AND." Additionally, we incorporated common meta-heuristic approaches, such as genetic algorithms, particle swarm optimization, artificial bee colony optimization, ant colony optimization, tabu search, and simulated annealing, to ensure comprehensive coverage in our search for relevant literature.

```
Str1:
[[Full Text: "procedural content generation"] AND
[Abstract: "Serach Based" ] AND
[[Full Text: "games" ] OR
[Full Text: "Computer games"] OR
[Full Text: "Video games"]] AND
[[Abstract: "Metaheuristics"   ] OR
[Abstract: "Meta-heuristics" ] OR
[Abstract: "heuristics" ] OR
[Abstract: "Inspired by nature"] OR
[Abstract: "Nature-Inspired"]] AND
```

```
[[Title: not["review"] OR
[Title: not["survey"] OR
[Title: not["overview"]]] AND
[E-Publication Date: (01/01/2007 TO 30/04/2023)]

Str2:
[[Abstract: "procedural content generation"] AND
[[Full Text: "games" ] OR
[Full Text: "Computer games"] OR
[Full Text: "Video games"]]
[[Abstract: "genetic algorithms"] OR
[Abstract: "particle search optimization"] OR
[Abstract: "artificial bee colony"] OR
[Abstract: "ant colony optimization"] OR
[Abstract: "tabu search"] OR
[Abstract: "simulated annealing"]
[Abstract: "cultural algorithm"]] AND
[[Title: not["review"] OR
[Title: not["survey"] OR
[Title: not["overview"]]] AND
[E-Publication Date: (01/01/2007 TO 30/04/2023)]

Str3:
[Abstract: "procedural content generation"] AND
[[Full Text: "games" ] OR
[Full Text: "Computer games"] OR
[Full Text: "Video games"]]
[Abstract: "Evolutionary"] OR [Abstract: "Evolution"] OR
[Abstract: "Physical Algorithms"] OR
[Abstract: "Swarm Intelligence" ]
OR [Abstract: "Swarm" ]
```

```
OR [Abstract: "Bio-Inspired"] OR
[Abstract: "Miscellaneous Algorithms"]] AND
[[Title: not["review"]] OR
[Title: not["survey"]] OR
[Title: not["overview"]]] AND
[E-Publication Date: (01/01/2007 TO 30/04/2023)]
```

(d) **Selection of Primary Research Study Sources**

The primary research study sources were selected in the fourth step of the planning stage. Our survey considered the most relevant database for this purpose. The sources included ACM, IEEE Xplore Digital Library, Springer, Elsevier, MPPI, and Hindawi. These venues provide a wide range of publications including book chapters, conference proceedings, and journal articles.

(e) **Inclusion and Exclusion Criteria**

The inclusion and exclusion criteria for the research papers were decided in the final stage of this process using the guidelines in Table 3.1 as a guide.

Table 3.1: The inclusion and exclusion criteria used in each research factor's methodology to evaluate each possible primary study.

Sr.No	Constraints	Inclusion Criteria	Exclusion Criteria	Justification
1	Period	Research published from to 2007-2023	Research papers published before 2007	By using this criterion, the evaluation is assured to focus on current PCG research, which is crucial for identifying current trends and improvements. It related to RQ1 and RQ2

Continued on the next page

Table 3.1 – continued from previous page

Sr.No	Constraints	Inclusion Criteria	Exclusion Criteria	Justification
2	Evaluation	Research includes PCG techniques particularly focusing on meta-heuristics	Research includes PCG using other techniques	This criterion ensures that the review focuses specifically on research that uses meta-heuristic approaches, which are related to RQ1
3	Purpose	Research contains generated game content using meta-heuristic approaches	Research contains generated game content using other approaches	This criterion ensures that the review focuses on research that is directly relevant to the research question, which is to evaluate the use of meta-heuristic approaches in PCG. It relates to RQ2
4	Comparison	Research involves a comparison between different meta-heuristic methods used in PCG	Research involves a comparison of other approaches	This criterion ensures that the review focuses on research that compares different meta-heuristic approaches, which is important for assessing their relative effectiveness. It is to support RQ2
5	Survey/ Overview	Research does not involve surveys/ overview/review of PCG techniques	Research involves surveys or overview of PCG techniques	Instead of simply summarizing the body of knowledge, this criterion ensures that the review focuses on research that offers thorough empirical data and analysis. This will not assist us in addressing our research questions.

Continued on the next page

Table 3.1 – continued from previous page

Sr.No	Constraints	Inclusion Criteria	Exclusion Criteria	Justification
6	Study	Research includes mathematical foundations, concepts, and experimental results in PCG or meta-heuristic methods	Research includes copyright and case studies in PCG or meta-heuristic methods. In addition, papers with languages other than English	This criterion ensures that the review emphasizes work that employs specific scientific procedures and analysis as opposed to just providing random evidence or case studies. Supporting RQ1.
7	Research databases	Papers are in research databases (IEEE, ACM, Hindawi, MDPI, and Springer)	outside the selected database	This criterion ensures that the review includes high-quality papers.
8	Duplicate Papers	Papers are not duplicated in different research databases	Similar papers in multiple research databases	This criterion ensures that the review avoids double counting or bias from including the same research more than once.

2. *Searching Stage:*

   An automated systematic search was performed following the procedure outlined in the previous stage. Figure 3.5 shows the search execution protocol.

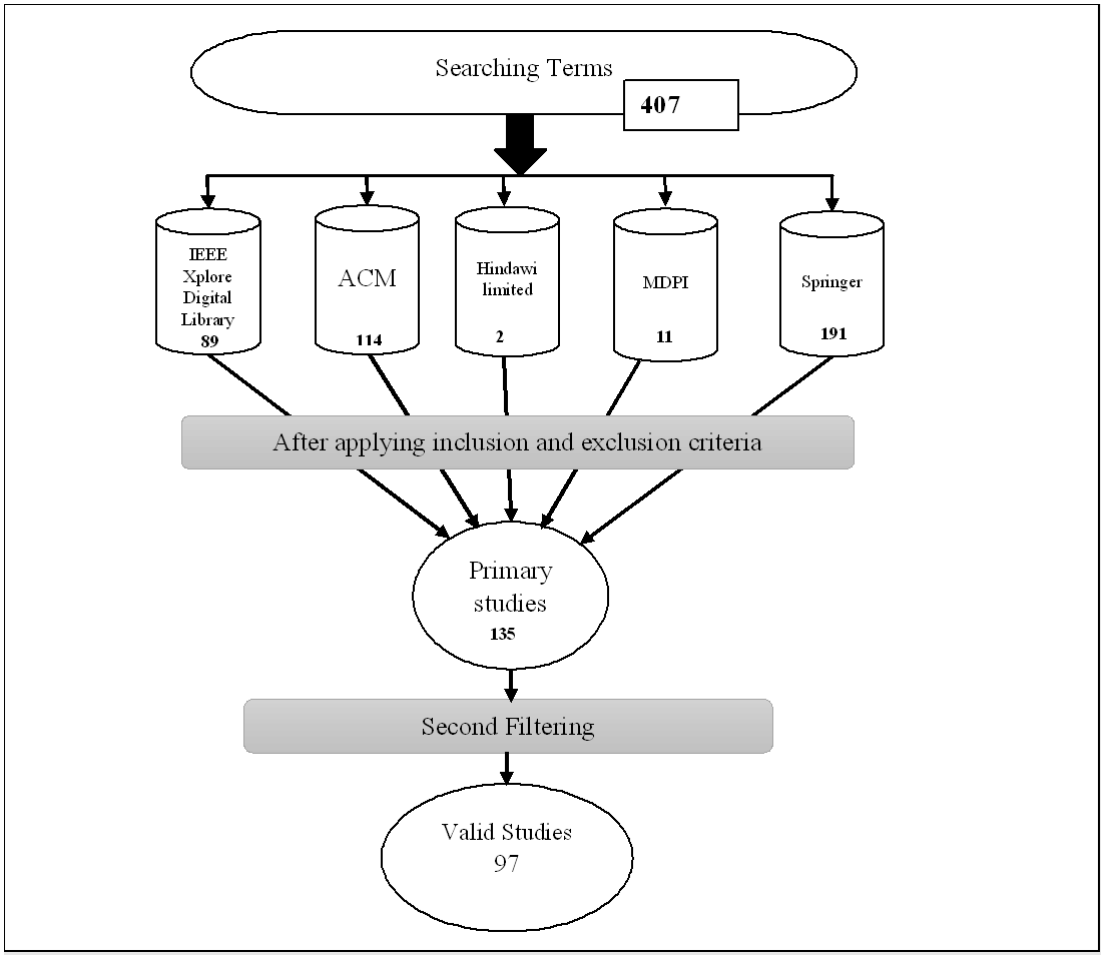


Figure 3.5: Search execution protocol

The primary goal of the systematic literature review is to evaluate the quality of recent studies on the topic. Although it is not possible to cover all existing literature, search terms were designed to retrieve most studies from internet databases. To address the challenge of poorly written titles and abstracts, additional steps are taken beyond reading titles and abstracts. Each paper's introduction, conclusion, and internal sections were carefully read to ensure a complete understanding of the contents of the research. Using these strategies, 407 primary studies were initially identified.

However, after applying additional criteria for inclusion and exclusion, 135 papers remained for evaluation. During the evaluation process, we considered both the relevance and research quality. Table 3.1 lists the various factors used to determine the relevance of a particular study. In cases where the abstract, introduction, or conclusion of a paper was unclear, further research was conducted to confirm the relevant keywords. Once we had selected and archived the papers, we conducted a second selection by reading the titles, abstracts, and keywords and assessing the usefulness of each study. Finally, we evaluated each research paper based on its relevance to the research questions. The quality of the assessment was based on the following criteria as seen in Table 3.2:

	Criterion	Description
QA1	Are the authors using any metaheuristic algorithm as part of the PCG approach?	It aims to determine whether the authors explicitly mention the utilization of meta-heuristic algorithm as part of their PCG approach of the reviewed papers
QA2	Are the authors solving the problem of generating game content using a meta-heuristic approach?	It focuses on determining whether the authors of the reviewed papers specifically address the problem of generating game content using a meta-heuristic approach. This assessment aims to identify whether the papers under review are dedicated to solving the challenge of creating game content through the application of meta-heuristic algorithms
QA3	Do the authors propose a new technique for performing PCG using a meta-heuristic algorithm?	It focuses on determining whether the authors of the reviewed papers propose a novel technique for performing Procedural Content Generation (PCG) using a meta-heuristic algorithm. This assessment aimed to identify whether the papers under review contribute to new approaches or methodologies in the field of PCG.

Table 3.2: The quality of the assessment criteria

During the quality assessment, we conducted a thorough examination of the selected research papers to identify any instances where the authors intro-

duced an innovative technique or methodology that utilizes a meta-heuristic algorithm specifically for PCG. Our assessment includes exploring whether the authors modified an existing algorithm, introduced a new algorithm, or combined multiple algorithms in a novel manner to address the challenges of PCG. Based on these criteria and the application of our quality assessment process, we identified 97 papers that were analysed in detail in this study.

3. *Reporting Stage:*

   The final stage of the review process involves reporting our research findings. This is discussed in the following section.

3.4 Results and Findings

In this section, we present the findings of the current literature review, which is organized into three subsections. First, in Section 4.1, a comprehensive overview of meta-heuristic algorithms in PCG is provided, offering a general summary of the research in the field. section 4.2 examines the analysis of the findings, addressing RQ1. Finally, in section 4.3, the focus shifts towards RQ2, providing specific insights and observations related to these inquiries.

3.4.1 Overview of Meta-heuristic Algorithms in PCG

A total of 407 publications were initially obtained when the search phrase was used in the specified databases, as shown in Figure 3.5. A thorough application of the inclusion and exclusion criteria was then conducted, leading to the elimination of several items that did not adhere to the predetermined criteria. Quality assessment was conducted to ensure the selection of high-quality and relevant research. A final batch of 97 papers was left after this careful process, and these 97 papers were thor-

oughly examined during the reporting stage.

Figure 3.6 shows a pie chart of the quality assessment results for the 135 research papers, after excluding 38 papers that did not met the criteria. It was sometimes difficult to determine whether the inclusion or exclusion criteria were met based on the title or abstract alone. As some papers that met the exclusion criteria were still included in the initial search, we forwarded them to the quality assessment phase for a more thorough review. The quality assessment results were as follows: 21 percent of the studies met QA1, 46 percent of the papers met QA2, and 13 percent of the papers met QA3.

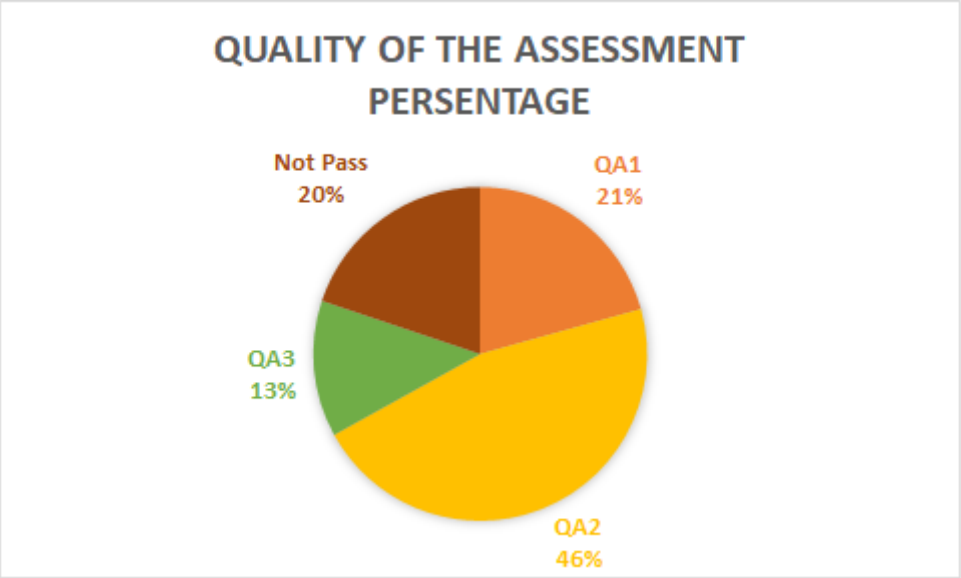


Figure 3.6: shows the percentage of the selected papers based on the quality

Figure 3.7 shows a graph of the number of papers published each year in the field of PCG, employing meta-heuristic methods from 2007 to 2022. There were no publications in 2007 or 2008, but the field gained its first research publication in 2009. Subsequently, the number of publications experienced periods of growth, with peaks in 2011, 2014, and 2021. The second year had the highest number of publications with 14.

This significant increase in the number of PCG papers in recent years can be attributed to several factors. The increasing popularity of video games and interactive media employing PCG, coupled with the development of more effective PCG algorithms, has contributed to this development. The exponential growth in PCG publications over the last decade aligns with the evolution of PCG approaches that are capable of generating complex and high-quality content. Figure 3.7 illustrates the dynamic expansion of PCG as a rapidly evolving research field. The increasing number of PCG papers underlines the growing interest in this domain and its potential to revolutionize the creation of video games and interactive media.

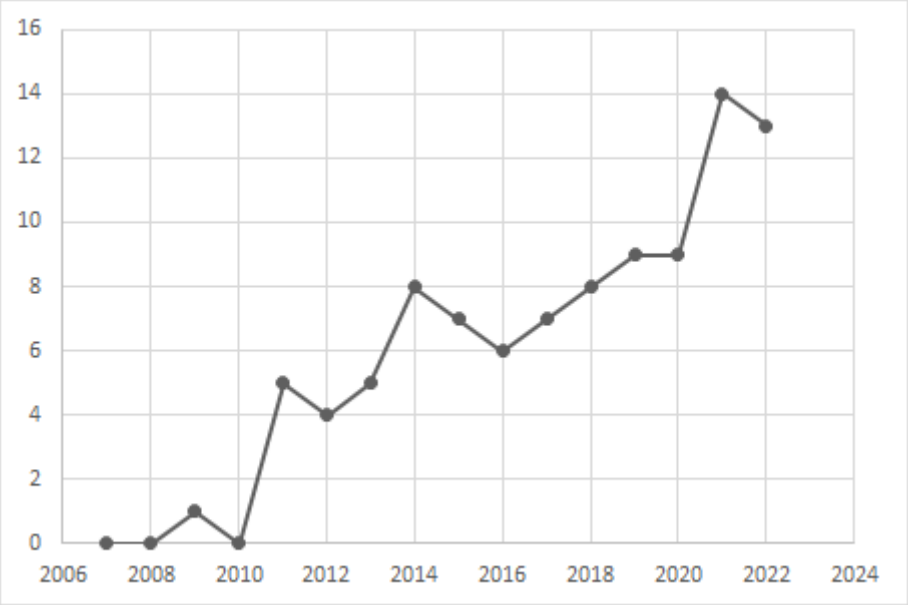
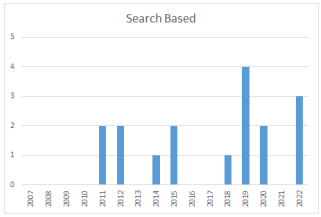


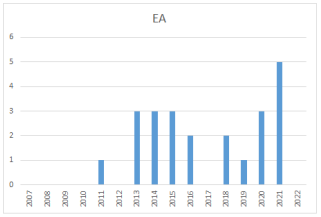
Figure 3.7: Displays the distribution of research papers over the selected period

Figure 3.8 provides visual representations illustrating the prevalence of various algorithms discussed in relevant published papers. In Figure 3.8a, the focus is on search-based algorithms, while Figure 3.8b highlights the utilization of evolutionary algorithms (EA), and Figure 3.8c specifically centers on genetic algorithm

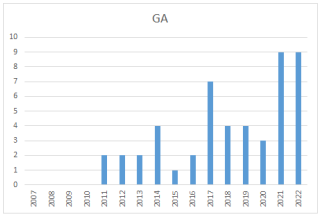
(GA). Additionally, Figure 3.8d pertains to particle swarm optimization (PSO), and Figure 3.8e is dedicated to simulated annealing (SA). These visual representations display a vital development in 2021 and 2022, when there was an increased emphasis on GA compared to search-based methods and EA. However, the other algorithms showed relatively limited usage during this period.



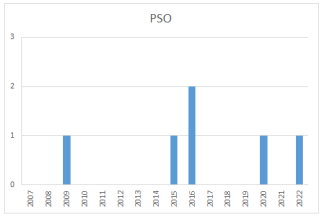
(a) Search Based



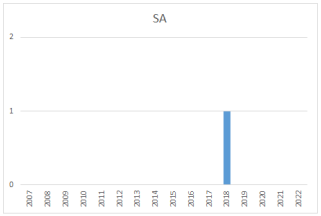
(b) EA



(c) GA



(d) PSO



(e) SA

Figure 3.8: Display distribution of using different algorithms over the selected period

As a starting point for using metaheuristics in PCG, the first emphasis was mostly on search-based strategies. Evolutionary algorithms or other stochastic search or optimization methods are used in search-based procedural content generation (SBPCG) to generate content that satisfies the necessary characteristics[23]. This method uses a test function, also known as an evaluation, fitness, or utility function, to score content based on a single number or set of related values. Using the findings of the test function as a foundation, new candidate content that may be of better value is generated algorithmically.

In addition, during our examination of the five papers published in 2011, two of them used GA to generate game content, while another pair explored search-based

techniques that incorporated dynamic programming as a fundamental component [74], [75], [76], [77], [78]. Additionally, one study focused primarily on EA. What is particularly compelling is a common theme that connects these papers. It is a shared emphasis on track race, tile, or maze generation, specifically aimed at enhancing the design elements of level generation.

These studies collectively employed a consistent methodology involving systematic exploration within the space of individuals, which often involves various design options. This approach may closely mirror the creative process employed by designers, who frequently combine diverse ideas to craft innovative designs that inherit desirable features from their predecessors. In essence, GA, EA, and search-based techniques closely align with designers' intuitive thought processes.

What further enhances the appeal of these algorithms is their seamless integration into the scope of gaming-level design. They effectively replicated the creative processes inherent in designing computer games. This strategy maximizes the exploration of the content space and significantly updates the generation of fresh and promising designs. The versatility of these algorithms could make them highly adaptable to the specific needs of procedural content generation (PCG), partially accounting for their common adoption within the PCG community.

Over time, SBPCG exhibits wide-ranging applicability, enabling the construction of various game content types, including puzzles, tracks, levels, terrain, maps, rules, mechanics, weapons, buildings, and camera perspectives. However, this method has certain limitations. SBPCG methods are generally slower because they evaluate numerous candidate content items. The successful application of evolutionary approaches relies on careful choices regarding the search algorithm representation and evaluation function [23].

The evaluation of game content quality poses the main challenge in search-based PCG, and is typically addressed through fitness functions. These functions can be classified into three key categories: direct fitness functions, simulation-based fitness functions, and interactive functions [48], see Figure 3.9.

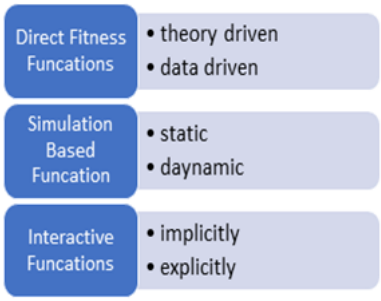


Figure 3.9: :Fitness Functions classification

SBPCG systems typically employ genetic and evolutionary algorithms. These methods have proven effective in generating both necessary and optional game content [79]. The following section discusses the dominance of these approaches over other metaheuristic techniques for PCG systems.

3.4.2 Findings (RQ1)

The research reviewed thus far in this work shows a clear bias towards evolutionary approaches to SBPCG. However, despite the apparent success of evolutionary approaches and genetic algorithms, there is no reason why game content cannot be generated by other meta-heuristic algorithms.

PCG has extensively utilized meta-heuristic algorithms to automatically create game content, such as search-based approaches, evolutionary approaches, and GA. These algorithms are frequently adapted to meet specific PCG requirements, such as designing game levels, generating textures, or creating characters. The ge-

netic algorithm (GA) is a frequently employed meta-heuristic algorithm in numerous computer game development contexts. For instance, Pereira et al. [30] presented a genetic-algorithm-based method for creating randomly generated dungeon maps with locked-door missions. This approach evolves a population of maps, selects the best maps based on the specified fitness parameters, and breeds them to create the next generation of maps. In a study involving 70 players, the researchers found that players preferred procedurally produced levels to their human-made counterparts in terms of difficulty, fun, and perception.

Similarly, Liapis et al. [80] used a genetic algorithm to improve the dungeon game levels by using sketches as concepts for different levels and generating high-resolution map segments. In Super Mario Bros, Ferreira et al. [81] introduced a multi-population genetic algorithm for PCG. It independently evolved four game aspects (grounds, blocks, enemies, and coins) and combined the best individuals from each component to construct a level. Furthermore, Togelius et al. [82] applied multi-objective evolutionary algorithms to generate complete and playable maps for real-time strategy games, thereby demonstrating the effectiveness of evolutionary algorithms in this genre. These studies exemplify the wide-ranging applications of meta-heuristic algorithms in PCG, with GA being particularly prominent in generating diverse game content across different genres.

In accordance with our prior discussion in Section 4.1, the initial preference for GA, EA and search-based approaches can be linked to their compatibility with the creative processes employed by designers when conceiving new and inventive game levels. This similarity may have served as a method prompting researchers to integrate these algorithms into the PCG domain. Over time, as researchers viewed the efficacy demonstrated by these algorithms in level generation, their application expanded to involve a wide range of content types.

SA has found limited applications in the field of procedural content generation (PCG), with its primary exploration occurring in a single study focused on gathering a diverse set of 2D/3D image data [83]. In this project, the SA approach was used during specific stages to create human-centric indoor scenes using stochastic grammar. However, the restricted use of SA in this project may not strongly motivate researchers to consider its application in a PCG context. One of the key factors that might contribute to SA's limited presence of SA in the PCG is its relatively slow computational speed. This characteristic is a common drawback of meta-heuristic algorithms, and can pose a significant limitation in PCG applications, where real-time performance is often a crucial requirement.

Another challenge associated with SA in PCG is the design of effective cooling schedules. Crafting appropriate cooling schedules is a critical aspect of SA implementation, and it can be particularly challenging, particularly when dealing with complex problems such as those encountered in PCG. Significantly, SA is a relatively general-purpose algorithm, which means that it can be employed to tackle a wide range of problems. However, this may not always be an optimal choice for each problem. In addition, the limited adoption of SA in PCG may be attributed to a lack of awareness of the algorithm among PCG researchers. SA is not as widely recognized as other meta-heuristic algorithms such as GAs and PSO. Consequently, PCG researchers may not be fully aware of the potential advantages that SA can offer for addressing PCG challenges and tasks.

In contrast, PSO has been discussed and utilized in PCG methods in five studies. The first documented use dates to 2009 when Hultquist et al. introduced their approach to generating parameterised procedural content using adjectival descriptors [84]. Their work emphasized that this approach was designed to complement

existing procedural techniques rather than replace them, offering a more natural method for procedural content generation. Through the incorporation of adjectival descriptors as an extra layer above procedural parameters, their approach sought to enrich conventional techniques and offer an alternative interface that proved especially useful for users during their initial learning stages. This initial exploration may partly explain why PSO did not initially receive extensive attention as a strong tool for content generation, with researchers focusing more on other techniques, such as GA and EA. However, this view may change in 2016, when Xia and Anand employed PSO for game content generation [85]. This marked a shift in perspective, recognizing the potential of PSO as an alternative approach for content generation. This significance is proven by the publication of two recent research papers in 2020 and 2022 [86], [87], further highlighting the growing recognition of PSO's capabilities in the field.

Furthermore, the existing literature provides limited evidence regarding the utilization of meta-heuristic approaches in PCG beyond classification of evolutionary algorithms, PSO and SA. As discussed in the background section, the literature encompasses a diverse range of meta-heuristic approaches, indicating potential opportunities for exploring alternative methods of PCG.

Table 3.3 presents a summary of meta-heuristic algorithms employed in the field of procedural content generation (PCG) research. It provides information on the frequency of use, specific applications, and initial release dates. Genetic algorithms (GAs) are the most preferred algorithms in PCG, appearing in 50 research papers. GAs are highly recognised for their ability to create customized content for players, including game levels, terrain, and items.

Search-based algorithms, such as the randomized prime algorithm, dynamic

programming, MAP-Elites (ME) algorithm, recursive backtrackers, grammatical evolution, novelty search, and estimation of distribution algorithm, have utilized evolutionary techniques to generate various types of PCG content. MAP-Elites and novelty search, two search-based algorithms specific to PCG, effectively produce diverse and high-quality content. MAP-Elites is used for designing platformers, racing tracks, and puzzles, whereas the novelty search contributes to the creation of Mario Kart tracks and platformer levels [88], [89], [90], [91], [92], [93].These algorithms enable developers to generate content that is exceptionally creative, diverse, and engaging. Moreover, they may be relatively easy to implement, making them an excellent choice for newcomers to PCG in the game-development community.

The table highlights the extensive use of meta-heuristic algorithms in PCG research, with GAs being the most prominent choice, although search-based algorithms offer unique advantages for generating specific content types, such as game levels or mazes. Additionally, the table emphasizes that the application of meta-heuristic algorithms in PCG research is a relatively recent development, with the first paper on the subject appearing in 2009, sparking growing interest in their use within this field.

On the other hand, EA has been employed in 23 research papers mainly for the creation of game levels, maps, and tracks. PSO was included in six research papers, contributing to game content generation, animation module development, and game balance enhancement. Conversely, SA has been utilized in a single study to facilitate game content generation. The table also shows that the number of PCG papers published in different categories varied over time. For example, the number of papers published on level generation has increased significantly in recent years, whereas the number of papers published on non-level PCG has remained relatively constant. This is likely due to the fact that level generation is a particularly challenging problem,

and there has been a lot of research in this area in recent years. However, there is a growing interest in non-level PCG, such as the generation of characters, stories, and music.

Meta-heuristic algorithm	Published research papers	Applications	First Release
Genetic Algorithm (GA)	50	Used to generate varied of game levels, and satisfied the players need	2011
Evolutionary Algorithms (EA)	23	Used to generate varied of game levels, maps and tracks	2011
Search Based	17	Used to generate varied of game levels and game contents, maze generation. Some are focused on optimizing the fun factor of the generated tracks	2011
Particle Swarm Optimization (PSO)	6	Game contents, animation modules, and game balance	2009
Simulated Annealing (SA)	1	Game Contents	2018

Table 3.3: Common meta-heuristics methods in PCG research

Nonetheless, Figure 3.10 illustrates the dominant position of GA with a usage rate of 52 percent in the field of PCG. Additionally, the figure presents the distribution of other algorithms, with search-based methods accounting for 17 percent, EA representing 24 percent, PSO contributing six percent, and SA accounting for one percent of the selected 97 research papers in this study.

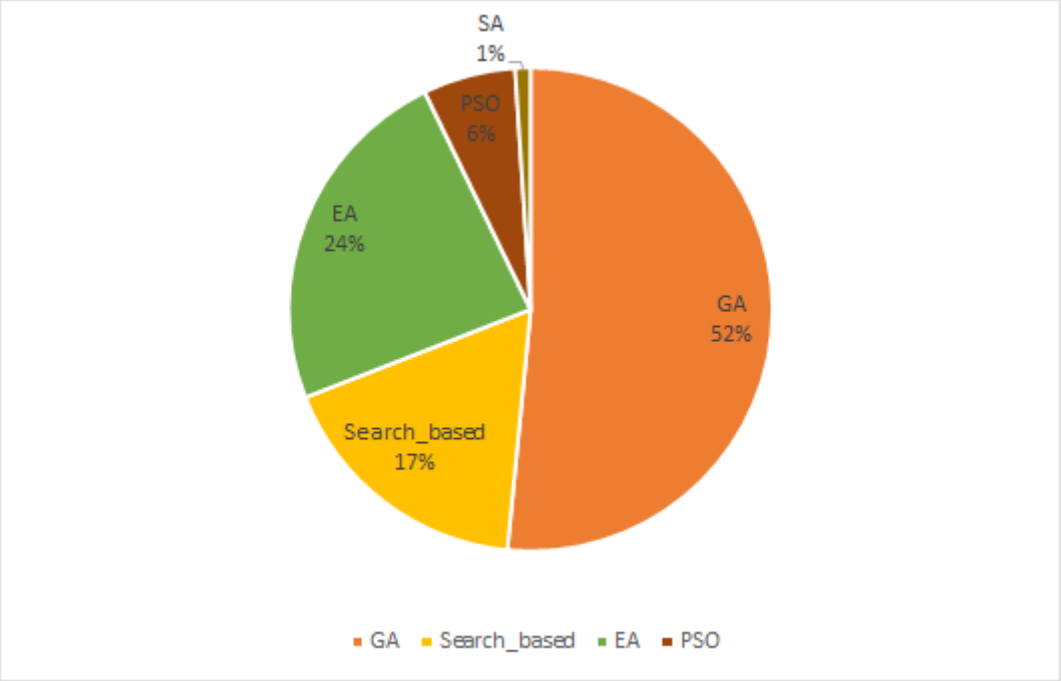


Figure 3.10: Show the percentage of applying individuals' meta-heuristics methods in PCG research

Considering that search-based and evolutionary algorithms have employed similar approaches to genetic algorithms (GA), it is evident that a substantial portion of research is dedicated to the utilization of evolution approaches in various contexts. This proportion reaches 93%, as shown in Figure 3.11.

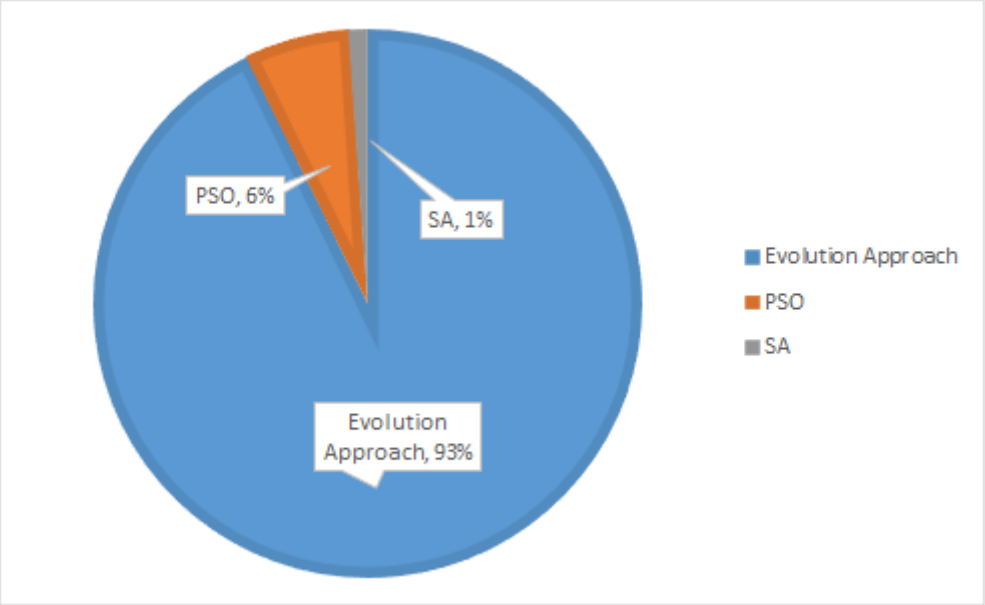


Figure 3.11: It shows that GA has a significant portion of the research in various contexts

3.4.3 Opportunities, Challenges and Future Research (RQ2)

We thoroughly examined 97 research papers cited in Appendix A and analysed various crucial elements. These include identifying gaps in research, assessing fitness and content quality, exploring challenges, and suggesting future areas of study. The investigation of using meta-heuristics in procedural content generation (PCG) yielded important findings, addressing both future research possibilities and current obstacles.

1. Research Gap in Exploring Other Meta-heuristic Approaches in PCG:

   Despite the effectiveness of individual meta-heuristic algorithms in PCG, there is an obvious gap in the research that calls for the exploration of alternative approaches. While these established algorithms have shown promise and good performance, combining them has the potential to enhance the quality of the generated content. The field of PCG has great potential; however, there is

still room for advancement through novel meta-heuristic algorithms tailored for specific tasks and adapting algorithms to different types of PCG content.

One possible path to progress involves creating new metaheuristic algorithms specifically designed for PCG tasks. Current meta-heuristic algorithms may not always be suitable for the unique challenges of PCG. This opens up the possibility of hybrid algorithms outperforming individual algorithms in specific PCG scenarios. In some cases, meta-heuristic algorithms may cause a computational burden, which is especially concerning for real-time PCG. This presents a chance for hybrid algorithms to outperform singular supplements in specific PCG situations, such as combining the global search abilities of a genetic algorithm with the local search capability of a hill-climbing algorithm.

Furthermore, there is potential for investigation into customizing metaheuristic algorithms for various PCG content genres. Different forms of PCG content have inherent variations that can significantly influence the efficiency of the meta-heuristic algorithm. For example, the range of possibilities for generating a game level is often much larger than that for creating a single game item.

2. Importance of Fitness Function Quality in Meta-heuristic Algorithm Effectiveness:

The effectiveness of a metaheuristic algorithm in procedural content generation (PCG) is undoubtedly linked to the quality of the fitness function employed to evaluate candidate solutions. Within the realm of metaheuristic algorithms, the fitness function plays a pivotal role, serving as a guiding compass for assessing the quality of candidate solutions.

In our comprehensive survey of selected research papers, a significant finding emerged: only 27 percent of these studies have dedicated efforts to develop a numerical fitness function for the algorithms they employed. However, this finding is not to be taken lightly, as it emphasizes a crucial point. In addition, these studies have proven that the representation of the problem under study significantly influences the quality and effectiveness of the fitness function.

In contrast, the remaining 73 percent have opted for alternative evaluation methods such as simulation or interactive fitness functions. These insights highlight the need for future investigation in the field of PCG. Specifically, there is a critical need to direct research efforts toward the creation of more precise fitness functions. These functions should be capable of comprehensively capturing the desired properties of the generated content. By enhancing the sophistication of fitness functions, researchers can further advance the performance and applicability of meta-heuristic algorithms in the domain of PCG, ultimately leading to more effective content-generation strategies.

3. Algorithms Integration:

   Combining different meta-heuristic algorithms has shown potential for making the optimization processes more effective. When researchers combine multiple algorithms, they can use each of their strengths to address complex problems. This approach allows them to create solutions that are better than those achieved using a single algorithm. It is an approach to improve these algorithms and perform more optimization research. In our research, we learned that 1 percent of the papers we examined employed multiple metaheuristic approaches to create novel PCG methods. To put this into perspective, only

one paper within our survey utilized both PSO and ecosystem implementation models (EIMs) to enhance game balance [85]. This study highlights the significance of recognizing the challenges in aligning player preferences with in-game content and gameplay, particularly within intricate game environments. Achieving a balance between game behavior and content is a demanding task for developers, which has prompted them to develop their unique approach.

4. Exploring mechanisms within meta-heuristics:

   To enhance the diversity within meta-heuristic algorithms, it is essential to explore the mechanisms that facilitate the exploration of various regions within the search space and prevent premature convergence. Strategies and techniques can be developed to achieve this, such as incorporating adaptive mutation rates, utilizing diverse crossover operators, and implementing selection strategies that prioritize diverse solutions. Prior knowledge of the problem structure, constraints, and objectives can further optimize the performance of the algorithm. However, our research revealed that only 6% of the surveyed papers dedicated effort to developing such mechanisms within their approaches. whereas the majority leaned towards conventional methods.

5. User Experience and Engagement:

   According to the findings of our survey, only 24 percent of the 97 research papers included in this study integrated user feedback as a key input in their approach. These users can represent a variety of roles, including players, designers, or a combination of both, and their feedback plays a pivotal role in evaluating the quality of generated content.

   In contrast, the remaining 76 percent of the studies relied on simulation meth-

ods to obtain feedback through numerical metrics and comparisons with established benchmarks. These insights underscore the importance of considering user experience, engagement, and a user-centric approach for assessing content quality within the field of procedural content generation (PCG).

This highlights the need to explore the design of PCG systems that prioritize user preferences and playstyles, while concurrently evaluating content quality from a user-centric perspective. There is an obvious demand for the development of metaheuristic algorithms that efficiently integrate user feedback. Such integration can pave the way for the creation of adapted and personalized content that resonates with individual user preferences and requirements. This user-centric approach holds substantial potential for enhancing the overall quality and relevance of the generated content in PCG research.

6. Assessing Content Quality:

   The development of metrics for assessing PCG-generated content is crucial for objective evaluation and optimization. Our study found that only 48 percent of the selected papers evaluated their results using metrics or expert feedback, indicating the need for greater emphasis on rigorous evaluation methodologies in PCG research. Utilizing metrics and involving domain experts ensures systematic comparisons and validation, driving advancements in the field, and the responsible use of PCG systems.

7. Diversity of the content's generation:

   The diversity of the content generated by the meta-heuristic approach was examined in each research paper within the respective applications. Our analysis revealed that only 15 percent of the studies explicitly discussed the diversity

of the generated content in their approach. However, most studies focused on the ability to generate game content.

8. Ethical Considerations:

   Our study indicates that no research paper has addressed ethical considerations associated with Procedural Content Generation (PCG) systems. This is an important observation because PCG systems have the potential to generate content virtually indistinguishable from human-created information. Issues regarding ownership, authorship, and authenticity may arise in this context. Therefore, it is imperative to conduct further investigations and establish responsible guidelines to address these ethical concerns regarding PCG systems.

In summary, Figure 3.12 provides a chart representing the percentages associated with the various aspects discussed in this section within the 97 studies. Remarkably, none of these aspects surpassed the 49 percent threshold. This indicates the need for future research to address these challenges and bridge existing gaps in the field.

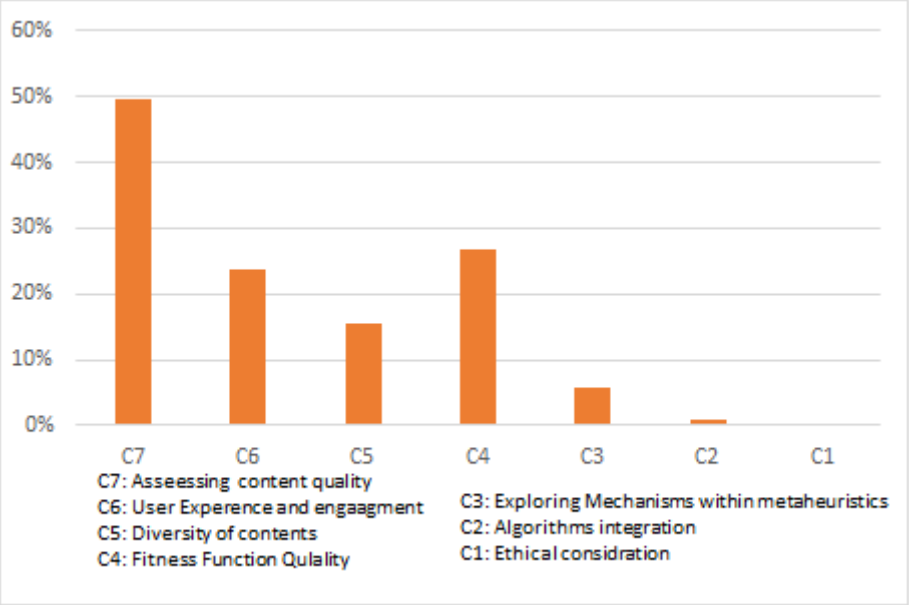


Figure 3.12: Provide analysis of different aspects based on percentage

3.5 Conclusion

This study adopted a systematic approach based on the methodology of Kitchenham and Charters (2007) [73] to comprehensively analyze the existing literature in the field. Relevant research papers were identified, classified, and analysed through a multiphase process of planning, searching, and reporting. This study evaluated the utilization of meta-heuristic approaches in generating game-play content and addressed the challenges associated with implementing these algorithms in PCG. It covers a wide range of meta-heuristic algorithms, including GA, PSO, and SA, across 97 selected studies. This article provides an overview of the literature review results, offers detailed insights into each paper, and addresses the two research questions.

The article first discussed the findings (RQ1) by examining the applications of different meta-heuristic algorithms in PCG and the algorithms that were most em-

ployed. This study revealed a strong focus on evolutionary approaches, particularly GA, in PCG. However, this bias towards GA does not negate the potential of other metaheuristic algorithms in generating game content. PCG has successfully utilized meta-heuristic algorithms such as GA, SA, and PSO for tasks such as level and map generation as well as game balance improvement. While GA demonstrates diverse applications in PCG across different genres, SA has seen limited usage, explored in only one study. PSO has been discussed and utilized in six research papers, focusing on game content creation, animation modules, and game balance. The existing literature provides limited evidence of alternative meta-heuristic approaches, suggesting the need for further exploration to enhance the diversity and effectiveness of PCG methods beyond GA.

The article concluded by discussing opportunities, challenges, and future research directions and answering (RQ2) in the field of procedural content generation (PCG) using meta-heuristics. This study highlights the need to explore alternative meta-heuristic approaches, enhance the quality of fitness functions, integrate algorithms, promote diversity, consider user experience, and establish evaluation metrics. Ethical considerations were also identified as crucial aspects that require further attention. The analysis reveals gaps in addressing these aspects, underscoring the importance of future research to advance PCG and bridge existing knowledge gaps. Suggestions for future research include the development of new algorithms, exploration of hybrid approaches, and examination of the ethical and social implications of PCG.

Overall, this article provides a comprehensive overview of the use of meta-heuristic algorithms in procedural content generation, particularly in computer games, and serves as a comprehensive reference for researchers and game developers intending to use optimization techniques to generate high-quality content in video games

and other digital media.

Chapter 4

Prelude Manuscript 2

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 the 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. Prelude Manuscript 2 focuses on comparing three metaheuristic approaches applied to race track games, with the specific goal of evaluating the effectiveness of different algorithms in producing game content. To this end, a GA, ABC, and 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 GA implementation. In general, this Manuscript is guided by research question RQ3 : ``**How do Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compare in terms of solution quality and convergence speed when applied to the task of race track generation in PCG?**"

Chapter 5

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

5.1 Introduction

Games are often considered a suitable test environment for artificial intelligence techniques, and game developers also focus on finding approaches that can automatically create computer game content to help minimize the load on developers [5]. Procedural content generated (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 [58] and has been effectively employed 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 has been employed [94], [5] and there is a growing interest in search-based approaches [95],[96],[35],[97]. The literature shows that evolutionary approaches, such as genetic algorithms, are one of the most widely used metaheuristic methods in search-based procedural content generation (SBPCG) and have been shown to be successful in solving a variety of complicated problems [50].

In other domains of study, comparisons of metaheuristic algorithm performance have shown that, in many cases, the popularity of genetic algorithms (GAs) is not entirely justified. While GAs excel in various tasks, they might exhibit limitations in specific contexts, such as slow convergence or becoming trapped in local optima. This motivates the exploration of alternative algorithms ([98], [99], [100], [101] Therefore, the main purpose of this study is to compare GAs to two different metaheuristic algorithms, namely Artificial Bee Colony (ABC) and Particle Swarm

Optimization (PSO), in the context of procedural content generation (PCG) for computer games. These algorithms have demonstrated promising characteristics in other domains, such as faster convergence for PSO and improved exploration of ABC. In particular, PSO offers advantages owing to its simplicity and ease of implementation. Additionally, PSO 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 [102].

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. Additionally, scout bees handle diversification by randomly generating new food sources, ensuring the algorithm doesn't 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 [103].

Specifically, this study provides an example of automatically generating race tracks for a racing game. The aim of this study is 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 paper is organized as follows. Section 5.2 provides an

overview of the literature supporting this study. Section 5.3 explains the methodological approach that includes the PCG task and implementation of the selected metaheuristics. The results of the comparison of the performance of the different search algorithms in the generation of race tracks for a racing game are presented in Section 5.4. The results and their implications along with future research are discussed in Section 5.5, and Section 5.6 concludes the study.

5.2 Background and Related Work

Procedural Content Generation (PCG) is the automated creation of game content through algorithmic processes [104]. Hendrikx et al. [19] proposed a comprehensive taxonomy that categorizes common PCG methods into six sections, as illustrated in Figure 5.1. While various projects in both the gaming industry and research have explored different methods for generating content across multiple levels of abstraction, this study specifically focuses 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 artificial intelligence category within Hendrikx et al.'s taxonomy, as 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 though a detailed examination of all the methods goes beyond the scope of this research.

In addition to Hendrikx et al.'s taxonomy, another significant taxonomy was developed by Zhang et al. [59], which classifies PCG methods into three main groups: search-based, traditional, and machine learning approaches. However, it is notable that meta-heuristic approaches were not explicitly addressed in their study,

potentially because of their categorization as a subset of search-based methods. This omission may stem from the observed similarities between the techniques employed in the search-based methods and meta-heuristics. Despite this gap, this study aims to contribute by focusing on a detailed comparison of evolutionary approaches in a broader context of search-based PCG methods.

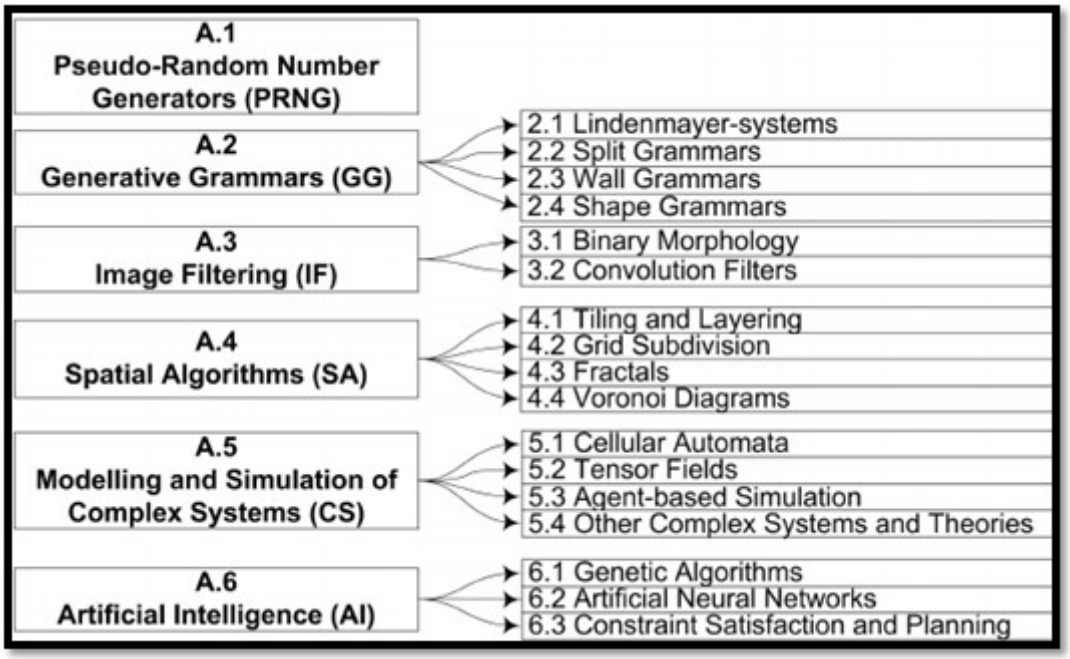


Figure 5.1: PCG Methods Taxonomy

Specifically, the algorithms selected in this study are classified as metaheuristic search algorithms. Togelius et al. [23] have described the concept of Search-Based Procedural Content Generation (SBPCG) as an advanced case of the more generalised "generate and test" approach. While the literature normally identifies it as a method that works on a population, as shown in Figure 5.2, this reflects the popularity of evolutionary algorithms, including genetic algorithms although it can also include approaches that are not population-based.

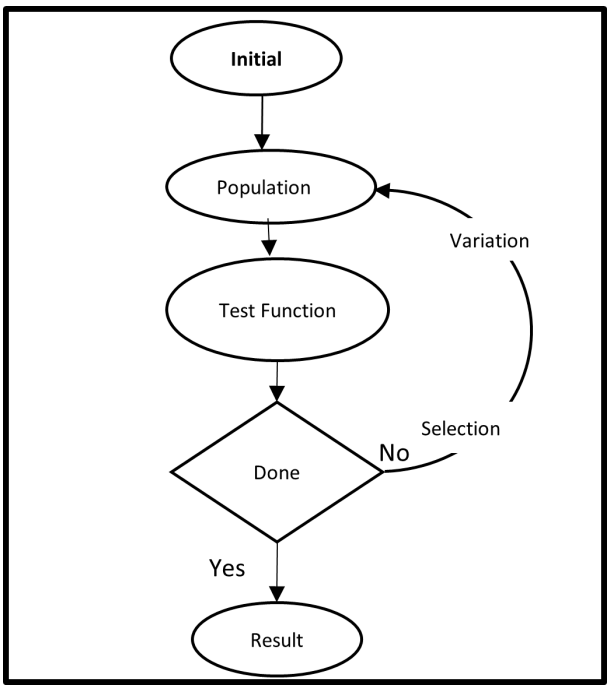


Figure 5.2: Search Based PCG

Such approaches have been extensively used in the literature. For example, Ferreira et al. [81] applied a multipopulation genetic algorithm 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 [105], puzzle games [106], nonplayer characters [107], game narratives [108] and crafted bosses in the Kromaia video game [95]. All these studies came to similar conclusions that procedurally generated content can meet the design intent of the game and contribute to the creation of playable levels.

However, given the apparent success of evolutionary approaches, there is no reason why game content cannot be generated by different algorithms. In many other domains, studies have shown that other algorithms can offer improved perfor-

mance, although some authors have indicated that there is a lack of standardization in the evaluation of algorithms [109], thereby emphasizing the importance of comparative studies in a given context. There is little evidence in the literature on the use of other metaheuristic approaches applied to the creation of content for games [110].

A study of both classical and current techniques has yielded numerous algorithms that can be applied to game content creation. Although the scope of the survey was broad, it did not consider that metaheuristic approaches could be used at all [111]. This suggests that the focus on metaheuristic approaches is relatively recent but does not explain the focus on evolutionary algorithms. Another review, which includes metaheuristic techniques, goes so far as states that SBPCG can utilize algorithms such as simulating annealing and particle swarm optimization [23]. However, the survey also did not provide any actual examples of their applications in PCG, and a direct search of the literature provides only limited evidence that alternative metaheuristics have been used in SBPCG. For example, Simulated Annealing and Markov chains [112], and particle swarm optimization has been used to automatically generate 2D graphical characters [113] as well as game levels for an infinite platform game [87]. Other algorithms have also been applied to gaming contexts. For example, Tabu Search has been applied to map generation for tabletop games [114] and race tracks [115]. There is limited research in the literature exploring the use of metaheuristic methods in computer games aside from evolutionary algorithms [50].

In the PCG literature, irrespective of whether a study is based on evolutionary algorithms or an alternative, few studies have compared the outcomes to those provided by any other meta-heuristic algorithm, nor is there strong justification for selecting these approaches based on an empirical understanding of their performance

characteristics. Some researchers have conducted comparative studies on the performances of metaheuristic algorithms. For example, a study conducted to generate race tracks in video games employed Tabu Search (TS) and GA using two different approaches. The results mostly show the variations between the two approaches in terms of the speed of convergence, where the GA is faster than the tabu search [115]. However, the 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 employed in SBPCG.

In contrast to PCG, the use of metaheuristic algorithms in other fields results in considerably more diversified representations of algorithms. For example, Dokeroglu et al. [109] listed 14 algorithms that have been used to generate new solutions to a range of problems; however, this list is far from exhaustive. Whilst genetic algorithms are still common in other domains, the use of other algorithms is much more prevalent and a range of comparative studies are undertaken in different areas over a long time period [116], [53],[117], [118],[119],[120].

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 procedural content generation in games. Metaheuristic algorithms belong to a domain of algorithms inspired by natural phenomena. These algorithms are also known as nature-inspired algorithms (NIA). This study does not depend on a specific classification, as some of them may belong to more than one category when selecting the algorithms for comparison. However, categorization depends on focus, emphasis, and perspective [121], [54]. 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 of the study were genetic algorithms, particle swarm optimization, and artificial bee colonies [122]. The latter two algorithms are examples of swarm algorithms in contrast to the evolutionary foundation of GAs.

Genetic algorithms (GA) are the first approach used in this study and are included as a baseline for comparison against which other approaches will be evaluated. It is the most commonly used evolutionary algorithm and is 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 modelling the genetic operators of crossover and mutation [123]. Over successive generations, the population evolves toward an optimal solution. Genetic Algorithms have been applied to a wide range of problems in operations research, engineering, and science [124],[125],[126], [127], [128],[129],[130]. Genetic algorithms have been widely employed in the computer game field as algorithms in search-based PCG, including generating context for massively multiplayer online games [131], levels for platform games [76], tower defense games [132], quest generation [108] and the design of levels in a dungeon crawler game [80].

In artificial bee colony (ABC) is the second metaheuristic approach utilized in this study 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 con-

sists of simulated honeybees searching for food sources [37]. Self-organization and the division of labor are two fundamental bases that are adequate for producing intelligent swarm behavior [133]. 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 the edge detection in digital image processing [67]. The development was specifically 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 presentation of the results and analysis of this method show that it is appropriate for solving antenna-design problems [68]. In the field of practical engineering optimization problems, ABC has also been used as an efficient approach to solve interval optimization problems. In this study, we developed a new method using ABC to produce additional perfect interval credibility [134].

In relation to PCG, the ABC algorithm has rarely been applied to video games. Mora et al. [135] make reference to the ABC algorithm, however in their study applied a different algorithm, namely the artificial flora algorithm. In another limited number of studies that include a comparative element, Sürer [136] compared the ABC algorithm to both PSO and the Firefly Algorithm in an abstract game, where the player has to rearrange daisies in a grid to trap 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, this study uses an algorithm for game mechanics rather than to generate specific game content, so it actually falls outside the PCG area.

Partial swarm optimization (PSO) also shows good performance in many appli-

cations, such as the ABC algorithm, which falls into the swarm intelligence 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 best position, and global best position accomplished by any particle in the search space. This iterative process continues until the best global position is reached [69].

PSO has been applied to a wide range of problems including robot path planning [137], ship design [138], and a limited number of applications related to video games [86]. There have been a limited number of applications of PSO as an algorithm in PCG tasks. For example, de Pontes et al. [87] 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 [139] 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. [140] suggested that there was minimal performance variance between GA, PSO, and ABC. However, ABC occasionally experiences delayed convergence, which contrasts with the findings of Karaboga and Akay [141], who demonstrated that the ABC algorithm surpasses other metaheuristics across various test functions. This discrepancy may be attributed to the particular problem studied by Kanović et al. [140].

In a study investigating GAs and PSO in evolving neural networks, Settles

et al. [142] observed that GAs perform better on large networks, whereas PSO performs better on smaller networks. Indeed, there is little convergence in any conclusion as to whether any algorithm is consistently better, and any outcome is highly dependent on the application domain and implementation.

Nevertheless, SBPCG studies 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. In addition, a few studies have attempted this type of examination using several algorithms to identify how they may appoint fulfilling PCG objectives. An investigation of other fields has shown that various algorithms have advantages over the genetic algorithm. Thus, this study is timely and relevant because it conducted an empirical evaluation of the different algorithms used in PCG applications.

5.3 Method

The purpose of this comparative study is to understand how well various metaheuristic search algorithms perform when utilized in PCG tasks to develop game levels. As highlighted in the literature review, there is a strong emphasis on adopting genetic algorithms 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 is quantitative, involving a comparison of the effectiveness and efficiency of the algorithms. These describe the ability to find decent solutions based on a designed fitness function and the convergence speed to a solution.

5.3.1 Research Design

In this study, a computational approach was employed to address the research objective. The primary methodology involves the application of GA, PSO, and ABC to a race track game. The process involves 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 employed an experimental design with a primary focus on determining significant differences between the algorithms. The experimental setup ensured that all the variables were held constant, with the only variable being the different algorithms used. This approach facilitates 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 conventional-sized 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,

particularly for intricate problems such as those we are addressing [143]. 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 to solutions. Therefore, smaller population sizes may be sufficient for PSO. While maintaining a certain level of diversity is crucial for effective exploration, an excessively large population may not be necessary [102]. Similar to PSO, ABC typically utilizes smaller populations and exhibits a rapid convergence. Consequently, a very large population is unlikely to offer significant benefits to ABC [103].

Recognizing the inherent stochastic nature of these algorithms, this study addressed potential performance variations by considering the average performance over ten 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 employing relevant analyses, as illustrated in Figure 5.3. 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.

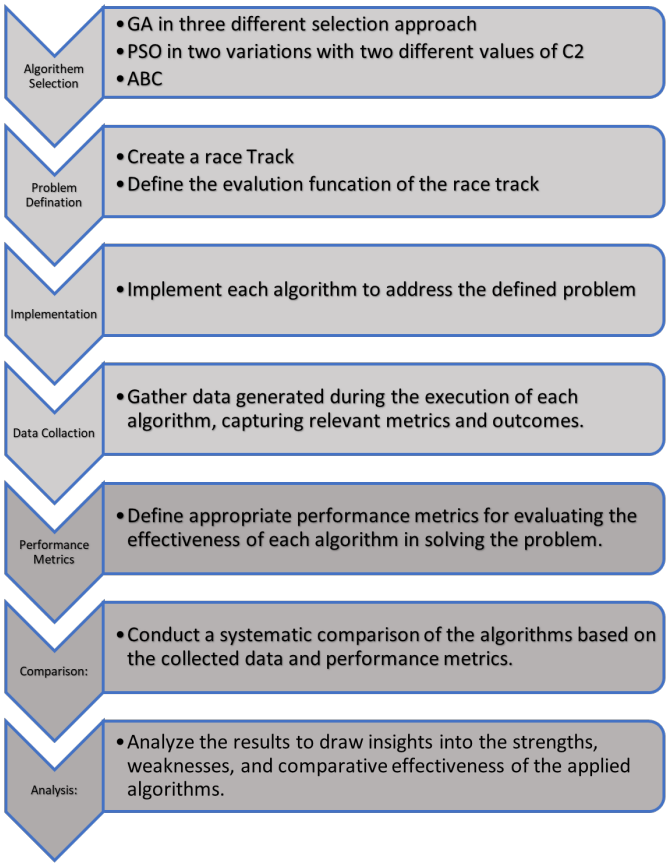


Figure 5.3: Research Design

5.3.2 Procedural Content Generation 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. Similar to the points on a circle, each point on the track has a distance (radius) from this central point, as illustrated in Figure 5.4. This figure highlights the importance of calculating the coordinates of each point using polar coordinates.

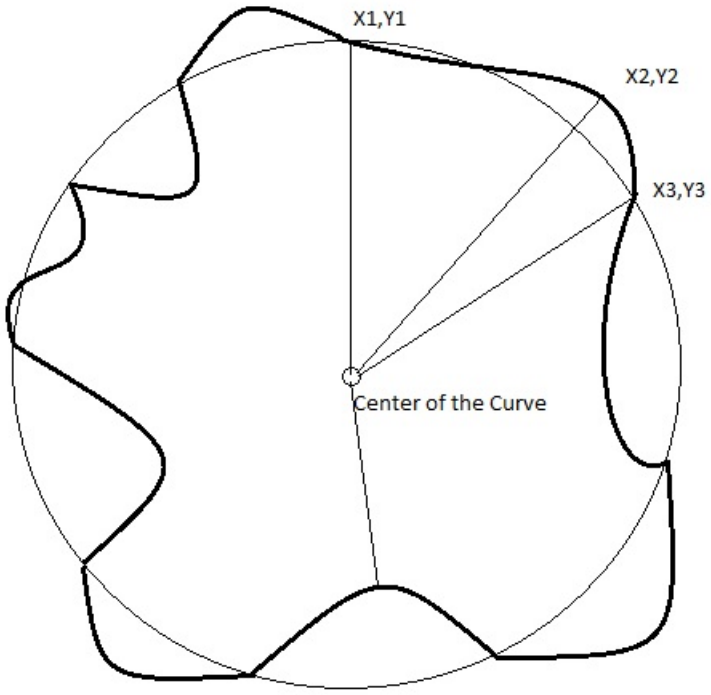


Figure 5.4: Visualization of a race track segment construction using polar coordinates.

As illustrated in Figure 5.4, points such as (x1, y1), (x2, y2), and (x3, y3) represent the connection points. Polar coordinates, which specify a point's location based on the distance from a central point (radius) and the angular direction, are ideal for generating these initial points.

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 determine the connection points between them. These connection points are crucial to ensure a continuous and well-formed race track. This was achieved by initially generating them using polar coordinates.

However, for our final track layout, we need connection points in terms of their x and y coordinates (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: the radial distance R_i and angular position ϕ. The radial distance signifies the distance of the point from a designated centered point (the origin), while 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 Fig. 5.5.

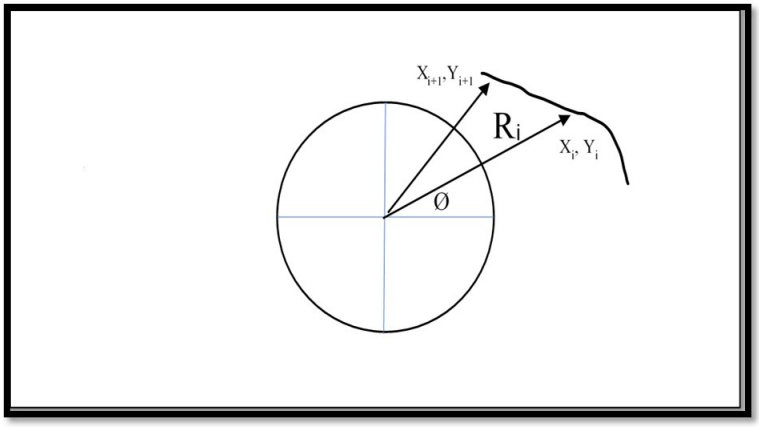


Figure 5.5: Generation of different points using a decreasing angle (ϕ) for each round.

To convert these initial polar coordinates (distance and angle) into the final Cartesian coordinates (x and y) used for the track layout, we employed well-established standard polar-to-Cartesian conversion equations 5.1 and 5.2:

$$X_i = R_i \times \cos(\phi_i) \tag{5.1}$$

$$Y_i = R_i \times \sin(\phi_i) \tag{5.2}$$

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 to 350 pixels from the center. This introduces a crucial element of randomness in the track layout. We chose 20° reduction because we had 18 seg-

ments. Dividing 360 °(full circle) by 18 points resulted in a 20° separation between each point.

Additionally, 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 each two neighbor points using Bezier curves [144]. This mathematical approach allows the construction of a closed loop representing the final race track.

Bezier 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, Bezier 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 based on 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 utilized cubic Bezier curves, which require four control points per track segment. These control points are denoted by P1, P2, P3, and P4 in Figure 5.6. The role of each control point in shaping the Bezier curve segment is 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 significantly influence its shape.

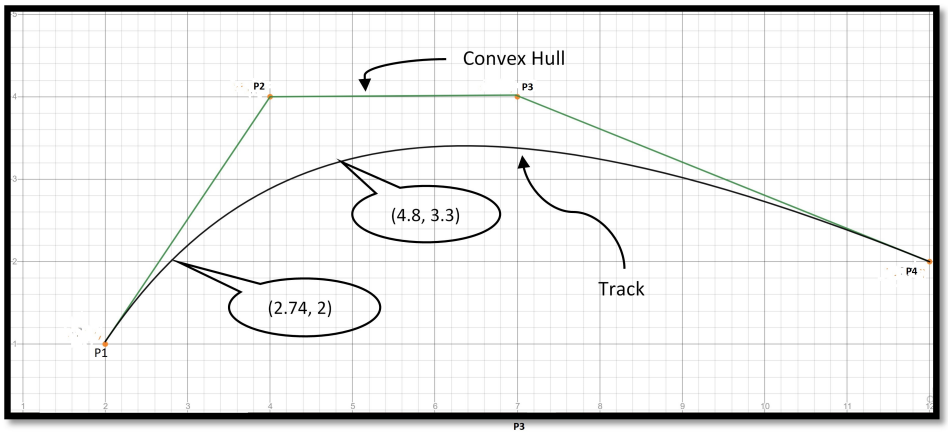


Figure 5.6: Generating different track shape using Cubic Bezier method

Although the concept of convex hulls can be relevant to Bezier curves in some advanced applications, particularly 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 Bezier curves typically involves ensuring that control points are positioned within the desired shape. By understanding the influence of each control point, we 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.

5.3.3 Evaluation (Fitness) Function

This study assumes that some track shapes are more desirable than others, although the intention of this research is 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 used in this study was relatively simple and aimed to direct the algorithms towards 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 [115] who considered both the curvature and speed profile as factors in optimizing race tracks.

As the intention of this work is to compare the algorithm performance and not find an optimal track, this has been simplified 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 [145], 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 5.7. Examining this figure, it can be seen that curve samples (b), (c), (e), and (f) present some challenges. In contrast, samples (a) and (d) appear similar to straight lines. This demonstrates why samples (b), (c), (e), and (f) were preferable for creating curved shapes.

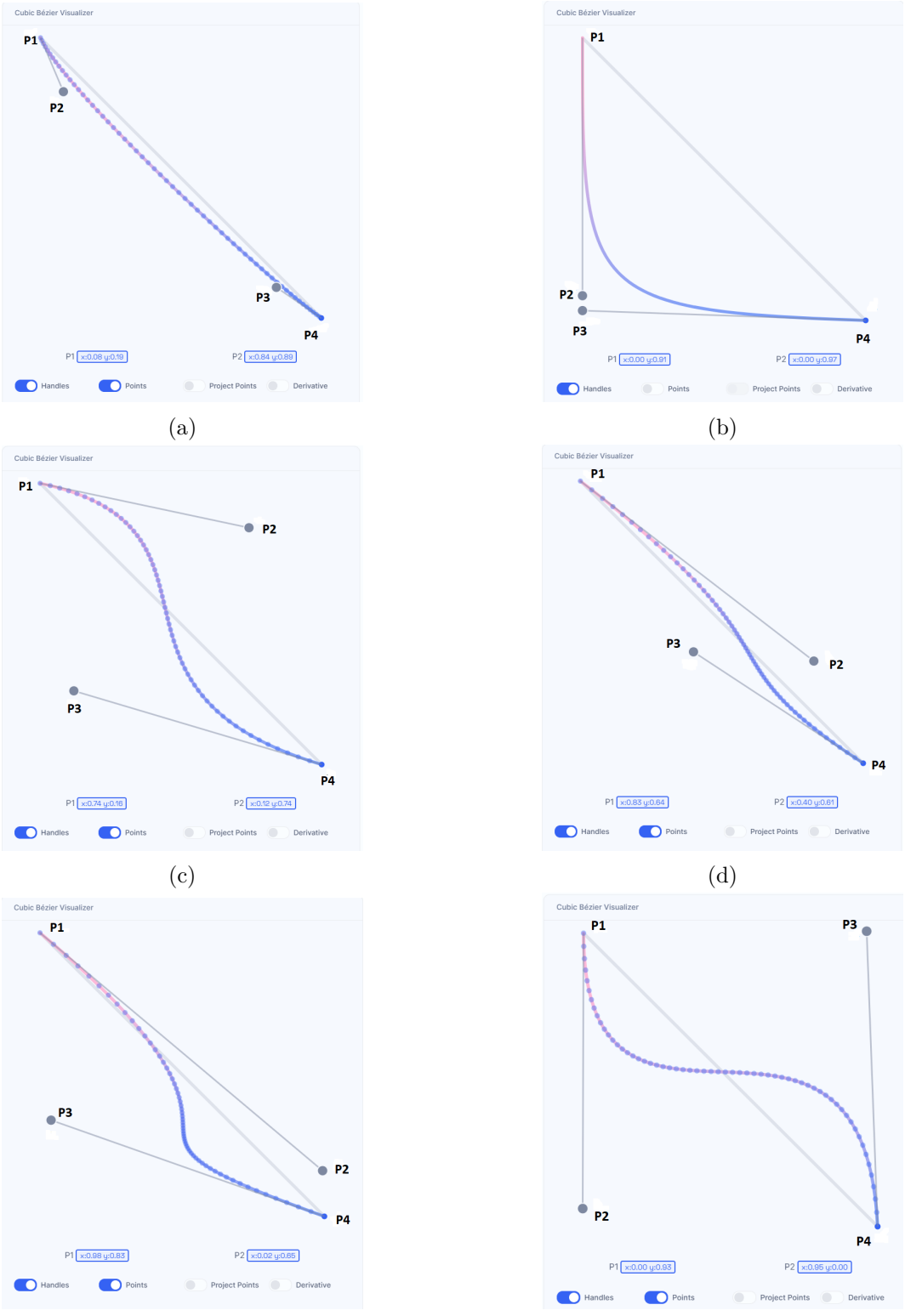


Figure 5.7: Different samples of Bézier Curve generation

Our analysis revealed that the distance between a control point and its neighboring segment edges significantly affects the scale of the resulting curve. Based on 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 5.8 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 Bezier method. In contrast, the control points labeled X1 and X2 represent positions that might generate undesirable curves.

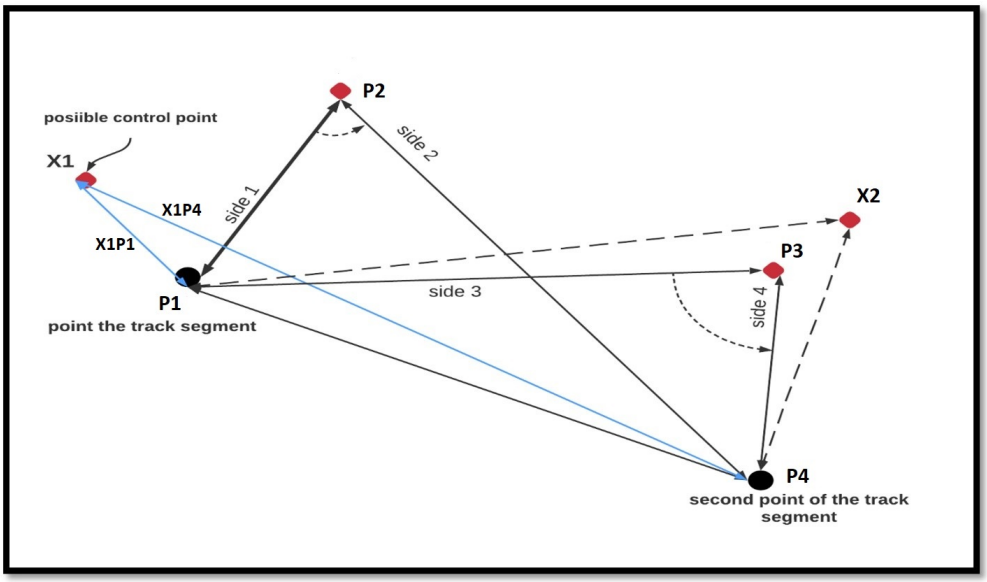


Figure 5.8: Describe the numerical scale of track segment

We calculate the distance between control point P2 and the first edge point P1 of the segment, denoted as side1. We then calculate the distance between P2 and the next edge point P4, denoted as side2. considering X1 as another possible position for a control point. We follow the same approach, calculating the distances

between X1 and both edges P1 and P4.

Our analysis revealed that the difference between side1 and side2 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 use the difference between these distances (subtraction results) for both control points (P2 and P3) or in cases with potentially bad curves (X1 and X2). Importantly, 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, it can be seen that the subtraction result of $\|\overrightarrow{P_1P_2} - \overrightarrow{P_2P_4}\|$ is less than that of $\|\overrightarrow{P_1X_1} - \overrightarrow{P_4X_1}\|$. Therefore, P2 is considered to be 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.

$$W_i = \|\overrightarrow{P_iP_{i2}} - \overrightarrow{P_{i+1}P_{i2}}\| + \|\overrightarrow{P_iP_{i3}} - \overrightarrow{P_{i+1}P_{i3}}\| \tag{5.3}$$

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

In summary, the fitness function is a mathematical formula that generates a numerical scale for each track segment, W, see Equation 5.3. The summation of these results is the total weight of the track, as shown in Equation 5.4. A track with a small total weight is preferable.

$$W_{track} = \sum_{i=0}^{17} W_i \tag{5.4}$$

5.3.4 Metaheuristic Implementation

The initial stage of the study implemented three metaheuristic approaches (GA, ABC, and PSO) and utilized these algorithms to generate race tracks to gain knowledge of their relative performance in producing game levels. However, the performance is determined in two dimensions: the effectiveness of the algorithms in terms of their ability to converge to a better solution in a fixed number of iterations and the consistency of this effectiveness across multiple attempts to generate race tracks. Figure 5.9 shows some typical tracks generated in this study, noting 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 using an improved fitness function in future work.

Figure 5.9: Sample of tracks from the initial population

5.3.4.1 Genetic Algorithm (GA) Implementation

The GA was the first metaheuristic approach used in this study. This consists of four steps after the initial population is generated. The first step involved selecting a pair of tracks (parents). However, the selection process uses three different approaches: a biased Roulette Wheel, Tournament, and a combination of the two approaches. The second step is the evaluation, which depends on the weight of the track. However, the track with the lowest weight was likely to be considered. The third step is the reproduction process, in which individuals' tracks pass their segments to the next generation. The last step is mutation, in which newly formed children are subjected to transformation. The genes were randomly selected for modification by exploring new values. Following the detailed steps, after the initial population is generated:

1. Selection of the Parents

   Our approach to parental selection within the genetic algorithm leverages three distinct techniques: biased roulette wheel selection, tournament selection, and 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 employed in genetic algorithms [146].

   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 served as a selection marker. The first track segment, whose corresponding section reaches the marker, is selected as the first parent. This process was 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. Consequently, this approach promotes the propagation of desirable track characteristics to the next generation, accelerating the evolutionary process towards optimal track designs. In the implementation, we used the following steps:

(a) Calculated the Total Fitness

(b) Calculated the proportion of each individual P_i according to the following equation 5.5:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{5.5}$$

Where N is the number of tracks in our population and f_i is the fitness value of each track

(c) Calculate the cumulative proportion CP_j of each track according to the following equation 5.6:

$$CP_j = CP_{j-1} + P_j \tag{5.6}$$

(d) Generate a random number R between 0.0 and 1.0

(e) Starting from the top of the population, the individual for which CP_j goes above R is the selected track (parent).

(f) The same process must be done in selecting the other parent.

The second selection strategy employed within our genetic algorithm (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 based on a competitive selection process [146]. Tournament selection involves creating a group consisting 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 two, essentially creating pairwise competition.

Following random selection, competition is conducted between K individuals. Fitness score, which serves as a measure of an individual's performance (track quality), was employed 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 towards populations comprising superior track designs.

Despite the previous selection methods, we incorporated a hybrid strategy that combined biased roulette-wheel selection and tournament selection as our third approach. During each iteration, we generated random numbers between 0 and 100 to determine the selection method. If the generated number is less than 50, we utilize the biased roulette-wheel method; otherwise, we use 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 was more likely to be considered for further processing and potential improvement.

3. The Reproduction step

   Moving on to the third step, we have 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 facilitated the exploration of different genetic combinations, as shown in Fig. 5.10.

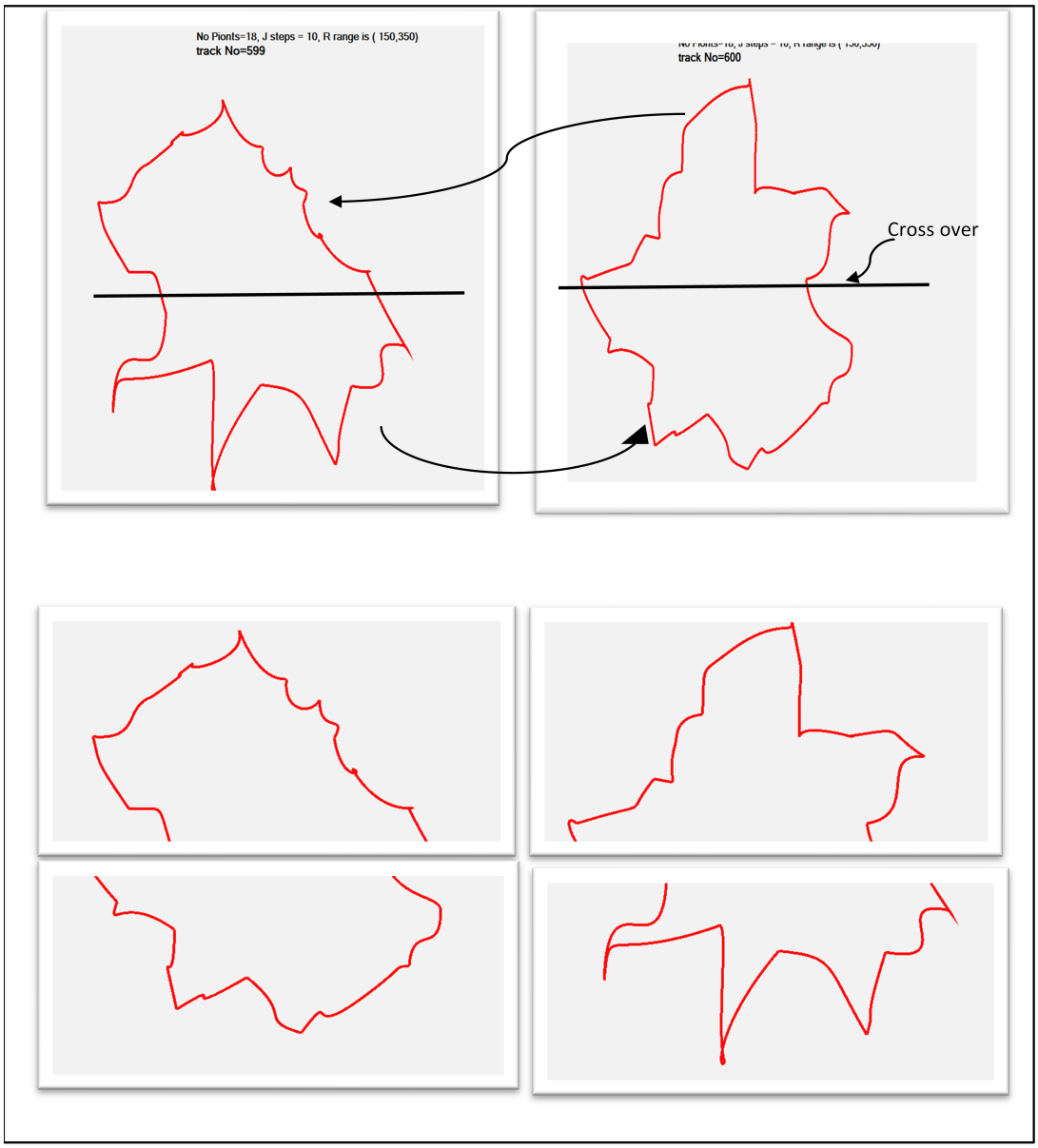


Figure 5.10: GA Crossover

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 were 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 provided an overview of the GA process used in our research methodology. The actual implementation on the CSharp platform involves additional details and specific parameters based on the research objectives and the problem domain, as shown in Figure 5.11.

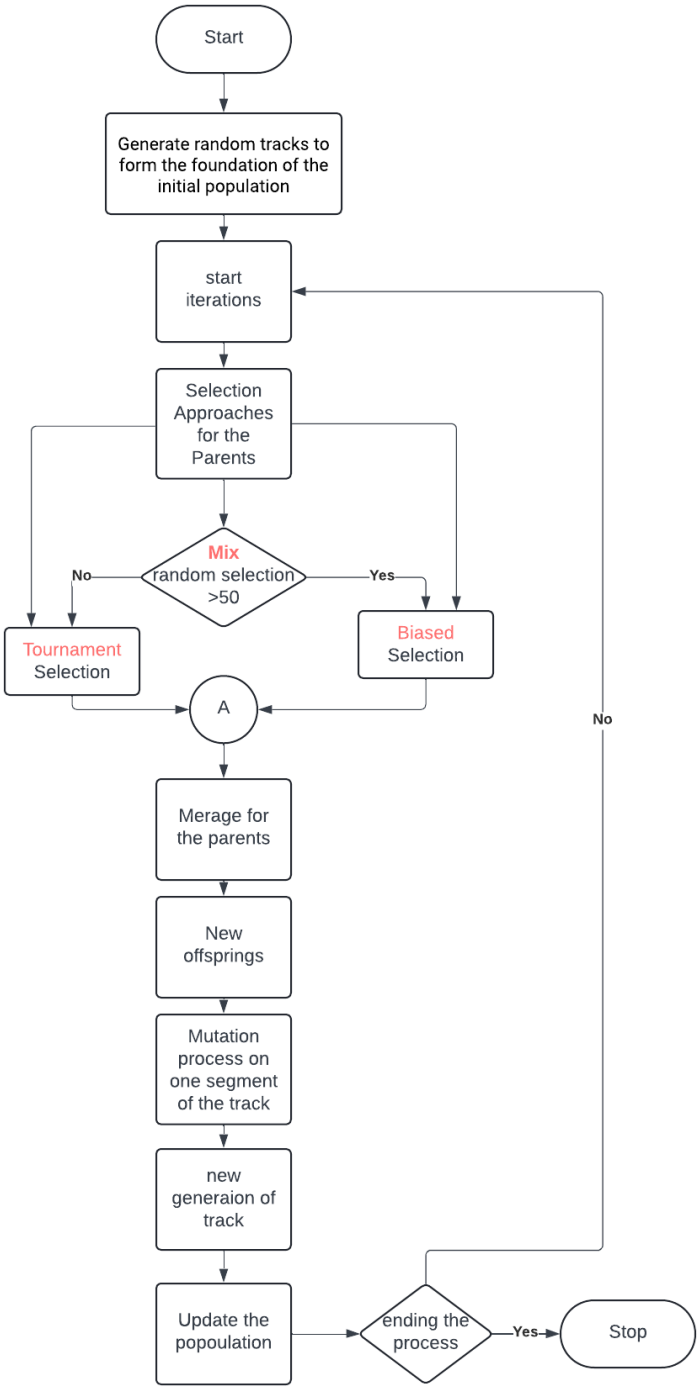


Figure 5.11: Steps of GA Approach in Tracks Generation

5.3.4.2 Artificial Bee Colony (ABC) Implementation

Our investigation extends beyond genetic algorithms, also exploring the potential of the ABC algorithm. This metaheuristic approach draws inspiration from animal behavior and the principles of swarm intelligence (SI) [37]. Swarm intelligence 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 [37].

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 utilizes 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 swarm intelligence[133], [147], are evident in natural honeybee colonies. These principles are further complemented by the satisfaction principles of the ABC algorithm.

While not as ubiquitous as GAs, the ABC algorithm remains a popular choice for solving diverse optimization problems. This study employed the ABC algorithm to iteratively generate an optimized race track through a structured three-phase process, excluding the initial phase.

1. Initial Population: During the initial population phase, 600 unique race tracks were randomly generated using the methods described in Sections 5.3.2 and 5.3.3. Each of these tracks, denoted by χ_i, represents a potential food source location within the ABC algorithm.

2. Employee Bee Phase: The second phase involves employed bees actively mod-

ifying 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 selecting 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 5.7. 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.

$$P(_i) = \frac{f_i}{Maxf} \tag{5.7}$$

where f_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 allowed the algorithm to expand its search boundaries and potentially discover superior solutions. By introducing new possibilities and exploring uninvestigated

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.

5.3.4.3 Particle Swarm Optimization (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. Notably, PSO has demonstrated computational efficiency, requiring fewer computations than GAs for problem-solving [148]. It is an efficient algorithm in terms of both memory utilization and computational speed [149].

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 [150]. The classical version of PSO determines a variable represented by a vector X as shown in Equation 5.8.

$$X = [X_1, X_2, X_3, \ldots, X_n] \tag{5.8}$$

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

In a swarm of P particles, each particle has a position vector X_i^t (Equation 5.9) and a population velocity vector V_i^t (Equation 5.10) at iteration T.

$$X_i^t = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in})^T \tag{5.9}$$

$$V_i^t = (V_{i1}, V_{i2}, V_{i3}, \ldots, V_{in})^T \tag{5.10}$$

These vectors are updated across the dimension j, which corresponds to the number of segments in each track. This update is governed by the equations. 5.11 and 5.12, where $i = 1,2,3,...,P$ and $j = 1,2,3,...,n$. In this context, each segment was updated based on the information obtained from Equation 5.11. The acceleration coefficient C_1 dictates the extent to which information is extracted from the best segment in the track so far in the iteration, while 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 5.12 However, C_1 assigned 2 for both versions C_2 are assigned a value of 4 for PSO#1, and a value of 5 for PSO#2.

$$V_{ij}^{t+1} = \omega V_{ij}^t + C_1 r_1^t (pbest_{ij} - X_{ij}^t) + C_2 r_2^t (gbest_j - X_{ij}^t) \tag{5.11}$$

In Equation 5.11, the parameters are defined as follows.

- ω is the inertia weight.
- V_{ij}^{t+1} is the updated velocity of track i in segment j at iteration $t+1$.
- 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.
- r_1^t and r_2^t are random values between 0 and 1 generated at each iteration to introduce stochasticity.

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \tag{5.12}$$

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, which is 0.3 in our case. However, the selection of ω may depend on the specific optimization problem and the behavior we want the algorithm to exhibit.

5.4 Results

The results presented in this paper aim to compare the performance of three different metaheuristic search algorithms on a procedural content generation task, specifically the generation of tracks for a racing game. Nevertheless, the results showed an evaluation of each selected metaheuristic method in terms of its efficiency and effectiveness. Usually, the fitness value of the tracks drops between 1600 and 100 points, depending on the scale considered in this study, as described earlier in Equations 5.3 and 5.4.

The maximum iteration number (200) and population size (600) are the same in order to compare 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 ten times. Consequently, the average of their outputs for each was considered for the study. Statistical information also includes the best values, mean values, worst values, and standard deviations.

5.4.1 Genetic Algorithm Application Results

The application of the GA is discussed in Section 5.3.4.1 and the results presented here relate to the performance of the three different selection approaches that have been implemented. Figure 5.12 shows the improvement in the fitness function over the 200 generations for which the algorithm was run using the mixed selection approach. Here, the line indicates the average fitness for each generation across ten runs of the algorithm. The error bars indicate the variation in individual runs relative to the average.

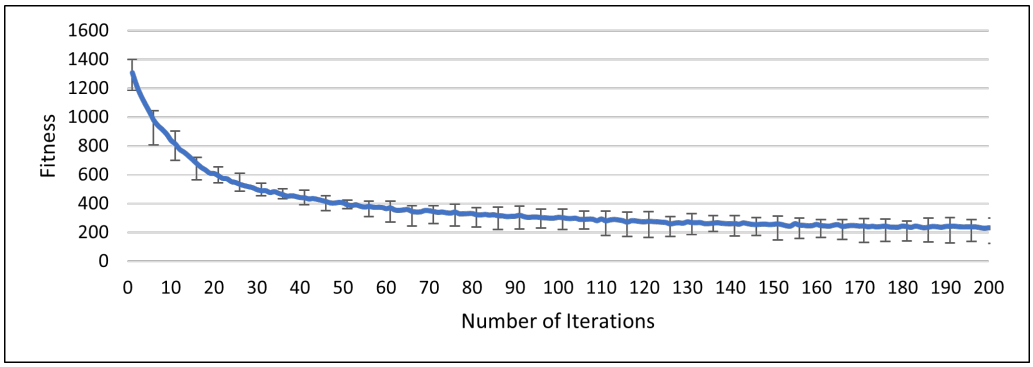


Figure 5.12: Genetic Algorithm Average Convergence (MIX)

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 1200-1400 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 5.13 displays the results of using GA with a biased selection method. While the initial average fitness is comparable to that in Figure 5.12, the initial convergence is slightly slower and the final fitness values are higher, with an average value of approximately 400.

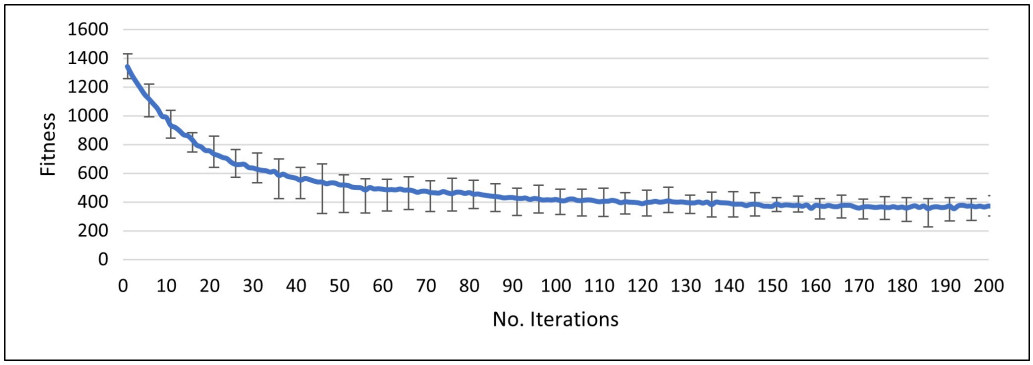


Figure 5.13: Genetic Algorithm Average Convergence (Biased)

Figure 5.14 shows the results for 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, in comparison to the 50 generations required using the mix selection approach. While the final average fitness shows no real difference from that achieved using the mixed selection method, the variability across multiple runs is significantly lower.

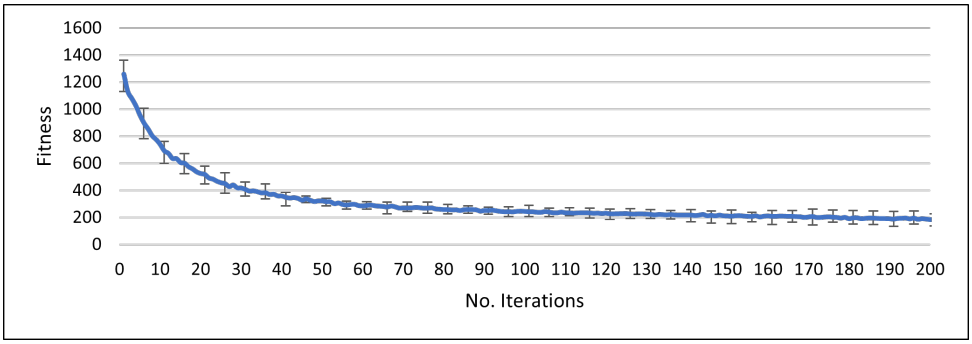


Figure 5.14: Genetic Algorithm Average Convergence (Tournament)

5.4.2 Artificial Bee Colony Application Results

Figure 5.15 shows 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 while 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 over 100 points. More importantly, the variability across multiple runs is virtually non-existent, implying that this performance is consistent and that multiple runs are not required to find a good solution.

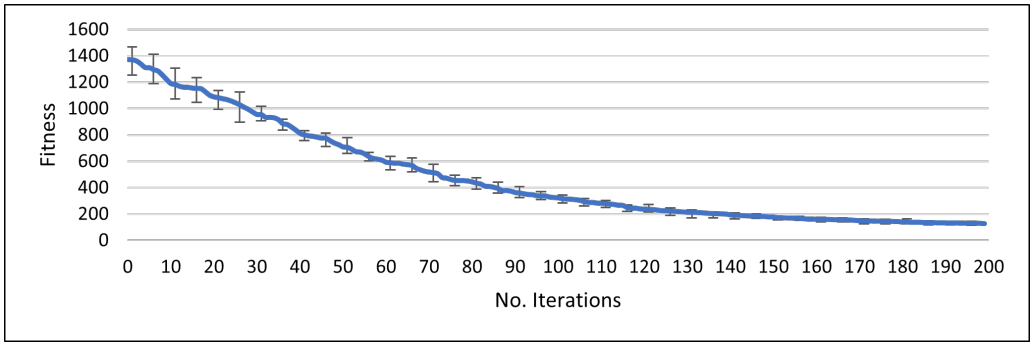


Figure 5.15: Artificial Been Colony Convergence

5.4.3 Particle Swarm Optimisation Application Results

Similar to the GA implementation, two variations of the PSO algorithms were implemented with two different values of the $C2$ acceleration coefficient in Equation 5.11, which influences the extent to which the search extracts information from the global position. For both implementations, the value assigned to $C1$ was 2. The first implementation, denoted as PSO#1, used a $C2$ value of four, and the second implementation, denoted as PSO#2, used a $C2$ value of five. Figure 5.16 presents the results of the first implementation (PSO#1). In comparison to both the GA and ABC results, PSO shows a very fast initial convergence, reaching the threshold fitness value of 400 points in just ten 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.

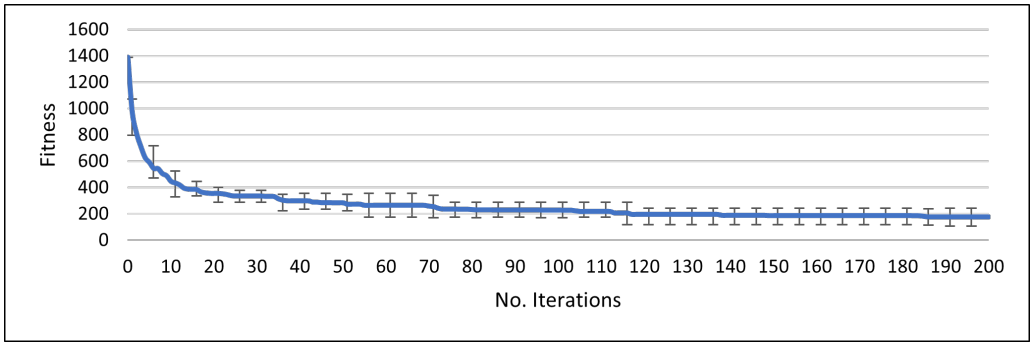


Figure 5.16: Particle Swarm Optimisation Convergence (PSO#1)

Figure 5.17 presents the results for the second PSO implementation (PSO#2), with a higher emphasis on global position information. These results shows little significant variation in the convergence behaviour, 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.

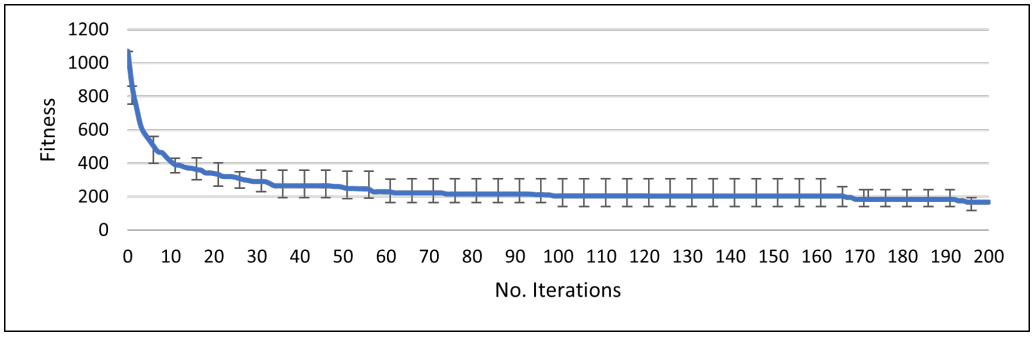


Figure 5.17: Particle Swarm Optimisation Convergence (PSO#2)

5.4.4 Metaheuristics Results Comparison

The results presented in Figures 5.12 to 5.17 provide a visual representation of the performance of the algorithms and an initial indication of their relative merits. Table 5.1 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.

Table 5.1: Comparison of different algorithms.

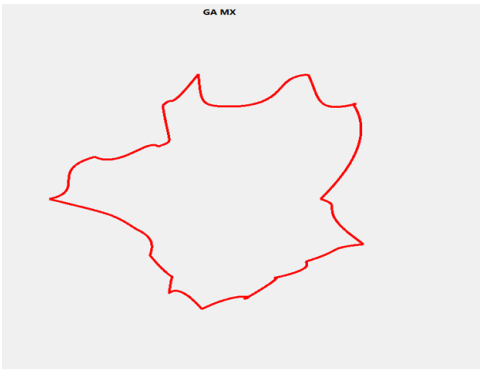
	GA(Mix)	GA(Biased)	GA(Tournament)	ABC	PSO#1	PSO#2
Worst	299.47	443.28	225.63	138.67	243.64	194.60
Best	121.75	304.30	137.74	113.98	106.88	115.86
Mean	233.09	370.1	182.9	125.9	175.5	166.4
SD	52.35	46.13	25.28	8.73	38.92	23.17

In relation to the best solution found by each algorithm across the ten runs, the overall best solution is found by the PSO#1 with a fitness value of 106.88. Comparing absolute values has little meaning in the context of the fitness function, as 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, particularly when compared to the biased selection implementation of the GA.

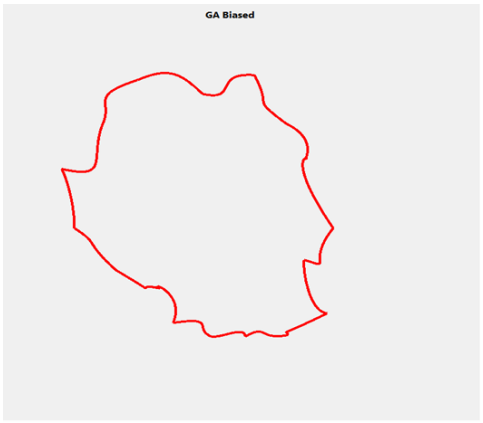
In relation to the best solution found by each algorithm across the ten runs, the overall best solution is found by PSO#1 with a fitness value of 106.88. Comparing absolute values has little meaning in the context of the fitness function, as there is no real means of determining whether a fitness value 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, particularly when compared with the biased selection implementation of the GA.

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 5.18 shows the best tracks produced by applying the GA,

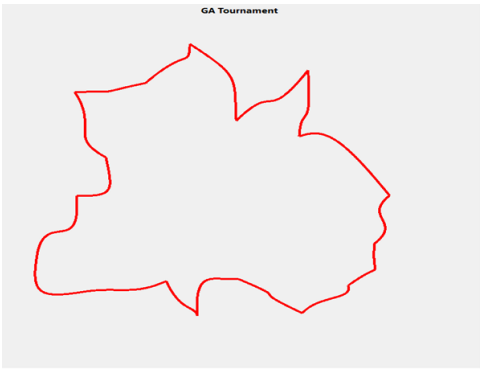
ABC, and PSO algorithms.



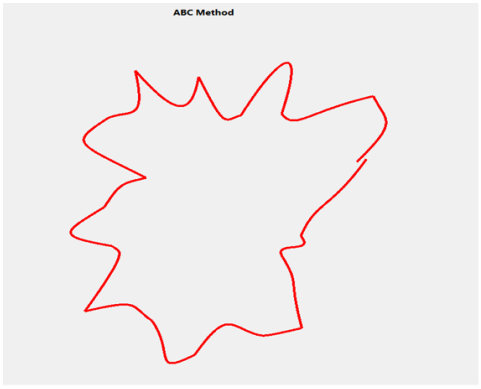
(a) GA Mix



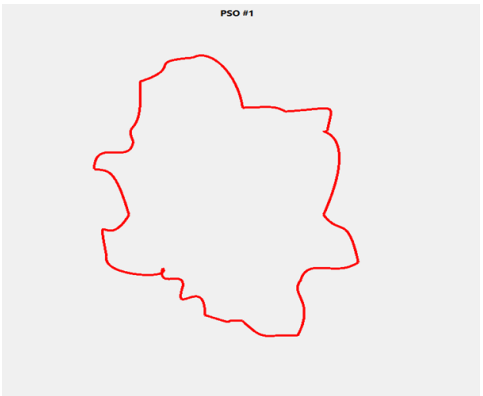
(b) GA Based



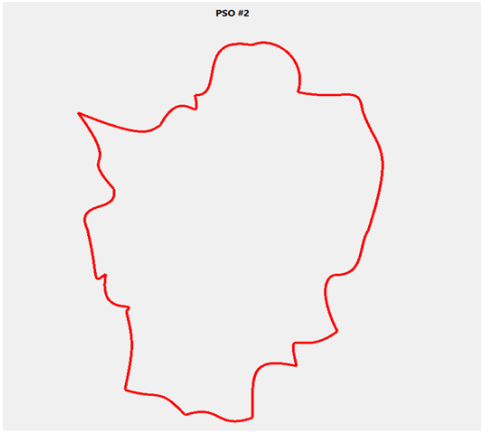
(c) GA Tournament



(d) ABC



(e) PSO Result 1



(f) PSO Result 2

Figure 5.18: Comparison of Resulting Tracks

Comparing the ABC solution with the solution produced by GA using biased

selection, there is a clear difference in terms of track curvature, with the ABC algorithm being significantly more curved. Clearly, race tracks 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 5.19 provides further insight into the performance comparison of the algorithms. Although ABC achieves the best results in terms of the mean value of fitness, it also exhibits the slowest convergence speed. Initially, the ABC algorithm showed slower convergence, but it significantly improved later and maintained consistent performance with minimal variability.

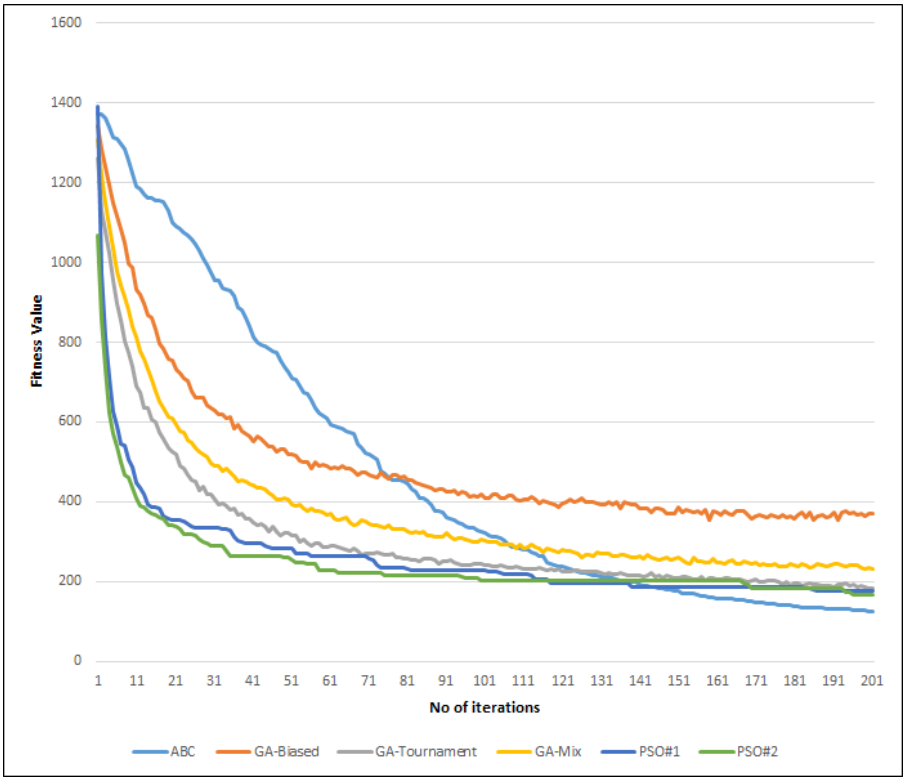


Figure 5.19: All the results for all the algorithms based on the average fitness

Figure 5.19 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. Initially, 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 both ABC and PSO appear competitive, based on their raw values. 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 the ABC and PSO belong to the swarm intelligence algorithm category. This common classification suggests that this type of algorithm is well suited to problems related to procedural content generation (PCG).

5.5 Discussion and Future Directions

This study investigated the performance of various computational algorithms for generating race tracks for video games. Our primary focus was to challenge the prevalent reliance on genetic algorithms (GAs) by exploring alternative methods. Our findings show that the selection strategy of GAs significantly affects their effectiveness. Selection methods prioritizing rapid convergence might initially identify suitable tracks but may limit the exploration of the design space, potentially leading to repetitive layouts. Conversely, methods promoting 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 the 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.

However, 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 utilizing 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 had several limitations and directions for future work. While a range of comparative studies exist in other domains, similar investigations are currently limited to procedural content generation 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 will 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.

Additionally, 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 a more systematic investigation of different population sizes and their effect on al-

gorithm performance, utilize the same number of evaluations for all algorithms to ensure fair comparisons, and incorporate statistical tests, such as ANOVA, 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.

5.6 Conclusion

This study measured and compared efficacious algorithms and their achievements in PCG tasks in various settings. GA, ABC, and PSO participated in this study to determine the best race track. In addition, the study tested GA using different selection approaches along with PSO in two different settings. However, this study developed a mathematical method that assumes an evaluation function to select the best track. However, the results were significant in showing that other metaheuristic approaches could considerably change search-based PCG. Despite the scope of This study contributes to the literature by expanding the knowledge of how different algorithms are suited to PCG. This could lead to 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 designer and player needs.

Chapter 6

Prelude Manuscript 3

PCG is an automated approach for generating game content using algorithmic techniques that offer advantages, such as cost reduction and time efficiency. This Manuscript aimed to evaluate the performance of meta-heuristic algorithms in PCG, specifically focusing on map layout generation. The manuscript challenges the assumption of the common use of GAs and provides evidence that other metaheuristic algorithms should be explored in PCG applications. Therefore, three meta-heuristic algorithms were compared to generate map layout: GA, PSO, and ABC, focusing on their effectiveness and efficiency in generating game levels.

Comprehensive experiments were performed using GA, ABC, and PSO for map creation tasks, and metrics such as convergence speed and content quality were used to analyze the performance and quality of the created content to identify the strengths and weaknesses of each algorithm in the context of game-level generation. The findings of this study suggest that GA employing tournament selection outperform other GA implementations, including GA with roulette wheel selection, a hybrid approach combining tournament and roulette wheel selection, an ABC, and two variants of PSO, when applied to generating game level maps. This reinforces the concept that the optimal metaheuristic algorithm for PCG depends on a specific task and the chosen evaluation methods. In addition, Manuscript 3 proposes that integrating alternative metaheuristic algorithms into PCG can enrich the field and inspire game developers to employ a wide range of techniques. Incorporating diverse meta-heuristic approaches enables game developers to enhance content generation and to create immersive, dynamic, and captivating game experiences. However, this manuscript assesses research (RQ4): " **In the context of map generation for PCG tasks, how are Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compared in terms of solution quality and convergence speed?".**

Chapter 7

Exploring Metaheuristic Algorithms for Enhanced Game Map Generation in Procedural Content Generation

7.1 Introduction

Games are often considered a suitable test environment for artificial intelligence techniques, and game developers focus on finding approaches that can automatically create computer game content to minimize the load on developers [5]. The procedural content generated (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 [58] and has been effectively employed 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. In the literature, a wide range of approaches has been employed [79] [5] and there is a growing interest in search-based approaches [96] [35]. Many researchers have employed evolutionary approaches such as Genetic Algorithms (GAs) in search-based PCG (SBPCG) because of their effectiveness in solving complex problems [50]. However, studies in other fields suggest that GAs may not always be an optimal choice [99] [100] [101].

This study builds upon our previous work [151] to further investigate this concept. In this study, the performance of GAs was compared with that of two other metaheuristic algorithms, Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO), in a new context: PCG for game map generation. Our prior research explored the performance of the same algorithms on a different PCG task [151]. The promising results achieved by PSO and ABC motivated us to investigate their effectiveness in another crucial aspect of PCG, map generation. By comparing these three prominent algorithms,

we aimed to gain valuable insights into their suitability for this specific task. Ultimately, our goal was to answer the following research question: "In the context of map generation for PCG tasks, how are Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compared in terms of solution quality and convergence speed?"

The remainder of this paper is organized as follows. Section 7.2 provides an overview of the literature supporting this study. Section 7.3 explains the methodological approach that includes the PCG task and implementation of the selected metaheuristics. The results of a comparison of the performances of the different meta-heuristic algorithms in the generation of game maps are presented in Section 7.4. The results and their implications are discussed in Section 7.5, and Section 7.6 concludes the study and discusses future research.

7.2 Background

Procedural content generation (PCG), which employs algorithmic capabilities to automate the creation of in-game content, has emerged as a critical field in video game development. The increasing complexities and costs associated with manual crafting of game elements have contributed to the growing importance of PCG. This adaptable technique is used across various genres, covering the automatic generation of game components such as levels, gameplay rules, textures, and in-game items [10], [5].

The origins of procedural content generation (PCG) can be traced back to the early stages of video gaming. In those early days, the PCG discovered its footing out of necessity, driven by memory limitations. Notable early examples include Beneath Apple Manor from 1978 [56] and Akalabeth, released in 1980 [15]. PCG have found complex applications, with a primary focus on en-

hancing replayability and adaptability. This imparts games with the capacity to yield vast pools of content, ensuring that each player's experience differs. As games develop into complex and modern forms, the challenge lies in generating content that combines diversity and quality seamlessly. Karavolos and Liapis's research paper examined the details of scaling PCG to meet the demands of modern gaming [152].

A diverse array of methodologies has emerged within the expansive domain of PCG, each offering distinctive advantages and use cases. Notable among these are conventional methods, search-based approaches, and machine learning techniques [5]. These methodologies collectively underscore the multifaceted nature of PCG.

Search-based procedural content generation (SBPCG), a subdivision of PCG methods, sets itself apart by its heavy reliance on search algorithms, frequently opting for evolutionary or metaheuristic search algorithms to navigate the complicated landscape of game content creation. SBPCG transcends boundaries and finds applications in diverse gaming contexts encompassing competitive and casual gaming experiences.

Metaheuristics represent a broad category of algorithms designed to address complicated optimization challenges. These methods are typically characterized by their iterative and stochastic nature, and incorporate randomness to navigate complex search spaces. Metaheuristics, such as those loaded with multiple constraints or subjected to noisy data, are highly efficient in addressing problems where exact solutions are elusive or impossible. A subset of algorithms inspired by natural phenomena exists in the field of metaheuristics. For instance, genetic algorithms are inspired by the principles of natural selection, particle swarm optimization takes prompts from the collective behavior of bird flocks, and bee colony optimization emulates the foraging behavior of bees searching for nectar. [153].

Metaheuristic algorithms have been explored to automate the generation of game content such as levels, maps, characters, and items. For example, the FI2POP genetic algorithm has been used to generate diverse and aesthetically pleasing levels for 2D games [35]. In another study, three general-level generators were presented and empirically compared [154]. In various other fields, research has demonstrated that alternative algorithms can yield enhanced performances. However, it is worth noting that some researchers have pointed out a lack of standardized evaluation methods for algorithms, underscoring the significance of context-specific comparative studies [109].

The current literature demonstrates a lack of empirical investigation into the application of alternative metaheuristic methodologies in game content creation. A comprehensive review of both classical and current techniques revealed a surplus of algorithms that are potentially applicable to game content generation. However, it is worth noting that this wide survey did not initially consider the utilization of metaheuristic approaches in this context [111]. This suggests that the recent focus on metaheuristic methods does not explain most evolutionary algorithms. Another thorough examination encompassing the realm of SBPCG revealed that simulated annealing and particle swarm optimization, among others, were used [23]. Furthermore, our independent investigation explored diverse algorithms employed in SBPCG, indicating a prevalent reliance on evolutionary algorithms, such as GA. However, the use of PSO and SA has been limited. This observation underscores the need for further exploration and research on the integration of metaheuristics into PCG, particularly those inspired by natural processes [50].

7.3 Research Approach

The primary objective of this comparative study is to provide valuable insights into the performance of various meta-heuristic algorithms when applied to procedural content generation (PCG) approaches for game-level design. Although genetic algorithms (GA) have received significant attention as an evolutionary strategy in PCG, it is crucial to explore alternative algorithms that can offer unique advantages in different application domains. This study places strong emphasis on quantitative evaluation, specifically focusing on the comparison of algorithmic effectiveness and efficiency, see Figure 7.1.

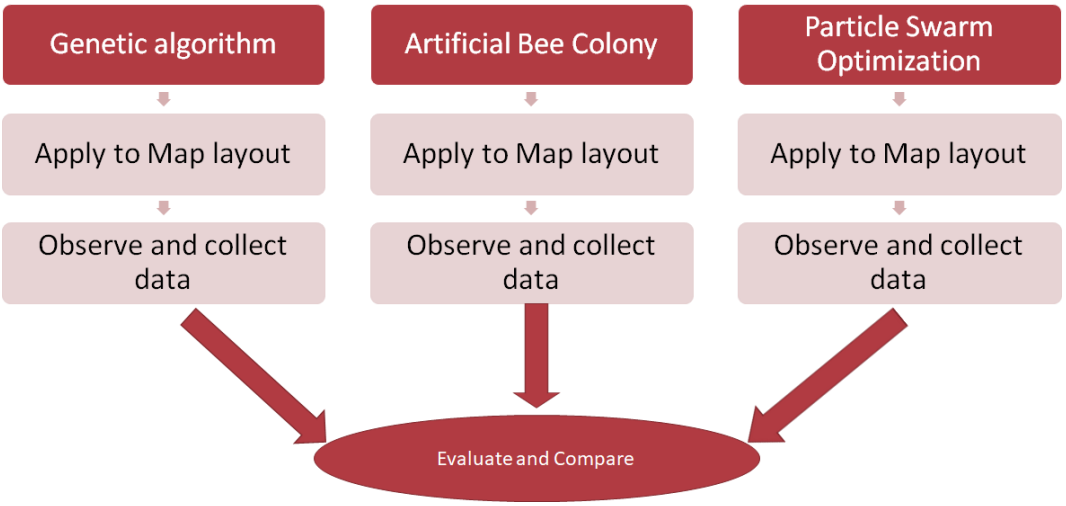


Figure 7.1: Research Approach

Efficiency refers to the speed at which the algorithms converge to a solution, whereas effectiveness concerns their ability to identify suitable solutions based on a predefined fitness function. It is worth noting that a qualitative or hybrid analysis incorporating playable interfaces and user testing would significantly enhance our analysis. However, implementing such a framework requires extensive preparation, including candidate purchases and the creation of qualitative/hybrid evaluation methods such as questionnaires and in-

terview scripts. This is a goal that we desire in future work to further develop the analysis of our methodology.

The research approach section is organized into four subsections, providing a comprehensive overview of our methodology. Section 7.3.1 presents an overview of the research design, outlining the key aspects of our research framework, an experimental evaluation, and an analysis of the algorithms. In Section 7.3.2, we discuss the selected procedural content generation (PCG) task and explain its specific characteristics and requirements. Section 7.3.3 introduces the developed fitness function and provides details about its formulation and role in evaluating the performance of the algorithms. Finally, Section 7.3.4 focuses on the implementation of the three algorithms (GA, PSO, and ABC), and describes the process of integrating each algorithm into our experimental setup.

7.3.1 Experimental Evaluation and Analysis of Algorithms

Because of the stochastic nature of the algorithms chosen in this study, the evaluation involved the creation of game maps and the application of relevant analysis to determine the significant differences that could occur between the algorithms. The study was effectively experimental, with all variables held constant except for the different algorithms that allowed the identifiable differences to be attributed to the change in algorithm.

According to the findings in Section 7.4, the focus of the study was not on the processing time and memory cost associated with applying the algorithms, nor did it require a significant amount of time or resources. The machine's specifications included an Intel(R) Core (TM) i59400F CPU running at 2.90 GHz, a speed of 2904 MHz, and a memory capacity of 16302 MB RAM. Information regarding the GPU is unnecessary because the procedure is performed on the CPU.

To ensure consistency in the comparison, each algorithm utilised the same population size of 600 individual solutions, and each was run for a fixed number of iterations, which were 200. However, the algorithms are inherently stochastic; therefore, to even out the performance of each algorithm, the study considered the average performance of each algorithm over ten runs. This study primarily emphasizes the quantitative approach, as depicted in Figure 7.2, and follows the steps outlined in the reference [155].

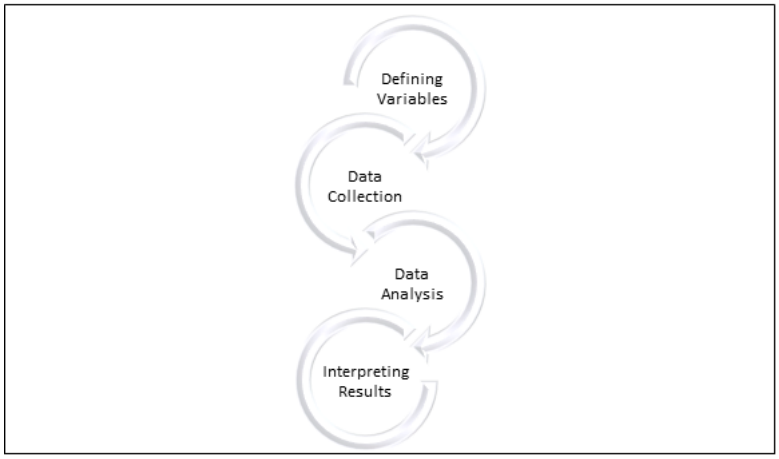


Figure 7.2: Our Quantitative Method

1. **Defining variables: We identified and clearly defined the variables of the algorithms that we wanted to measure or assess the efficiency and effectiveness of the algorithm. These variables must be numerically measured. Effectiveness is the highest scale and efficiency, which means that is the highest in iteration.**

2. **Data collection: Data were collected experimentally by running the algorithms in CSharp and Microsoft Visual Studio, and the collected data were in numerical form. The output was saved in an Excel spreadsheet.**

3. **Data analysis: After collecting the data, we employed statistical techniques. This analysis used descriptive statistics (e.g., average fitness,**

best fitness values, worst fitness values, negative range, positive range, and standard deviations) to summarize the data.

4. **Interpreting results: The analysed data were interpreted to draw conclusions and make generalizations regarding the algorithms under study. Statistical significance tests were performed to determine the significance of the observed findings.**

7.3.2 Procedural Content Generation Task

This study focuses on evaluating the performance of the algorithms by creating a 2D game-map layout. Although many strategy games offer hand-drawn maps for both single-player campaigns and multiplayer modes, there are compelling reasons for exploring automated map generation. One of the most apparent benefits is that generating a new map each time a game is played enhances its lifespan by providing players with fresh areas to explore and unique challenges to overcome. Moreover, this approach reverses the advantages gained by players in multiplayer games through map memorization [156].

Therefore, our proposed method involves generating a 2-dimensional NxM grid comprising 13 distinct tiles or components. Each of these tiles or components may have open doors on its sides. These doors provide players with the means to navigate through the map, and they play a crucial role in finding the way to the ending point, as shown in Figure 7.3.

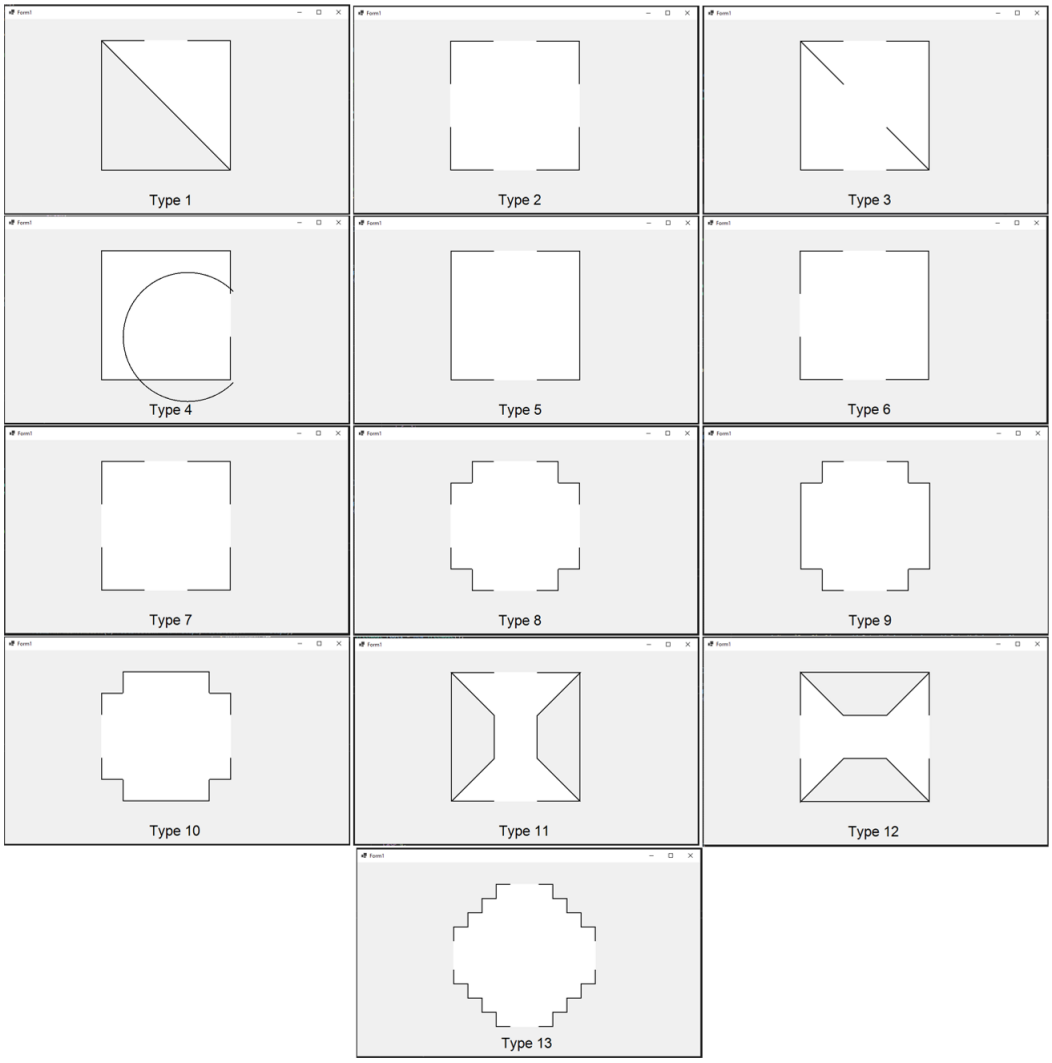


Figure 7.3: Different parts that make the map

As shown in Figure 7.4, we incorporated various tile types to illustrate the functions of the open and closed doors. This dynamic map generation not only introduces diversity and unpredictability to gameplay but also challenges players in adapting to different layouts and making strategic decisions based on the arrangement of open and closed doors.

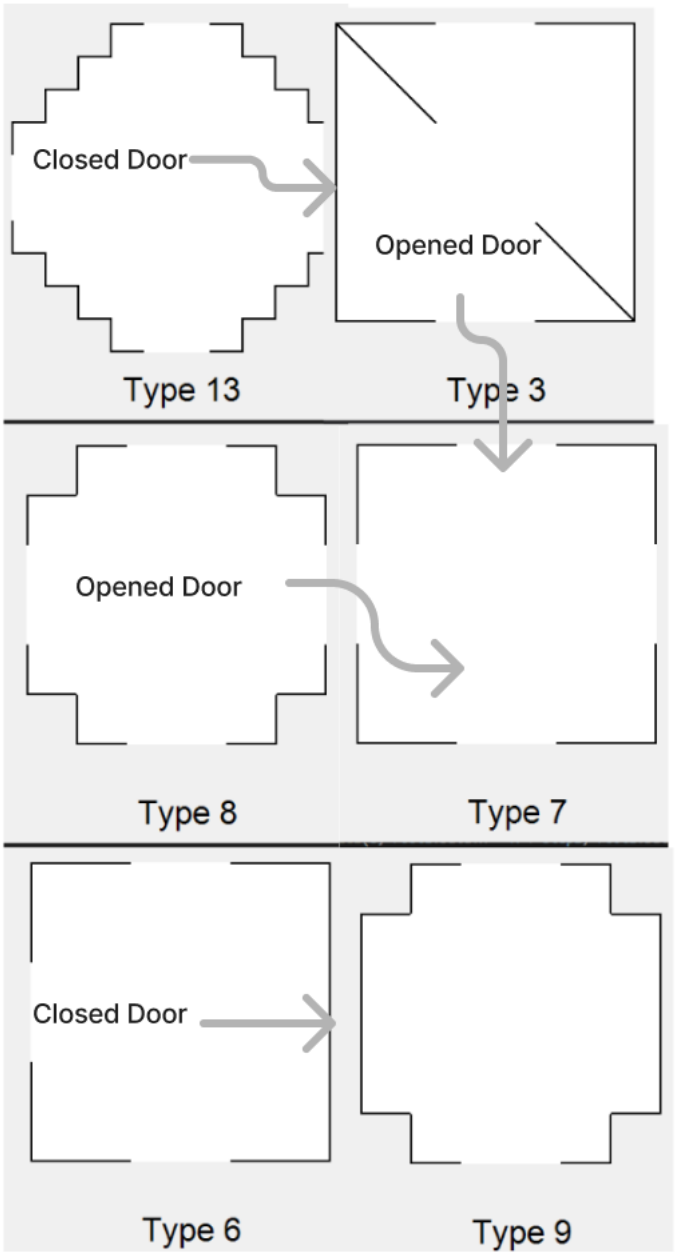


Figure 7.4: Closed door and opened door in the map

7.3.3 Evaluation Function

Our algorithms employ a fitness function that takes an individual, symbolizing a map in this context, as input and generates a numerical value that

reflects the evaluation of that map. In our approach, we assess the quality of a map by considering the challenges it offers in navigating from Point A to Point B. This notion parallels the exploration in a relevant research paper [41] where researchers aimed to accomplish specific objectives. Although our game presents multiple potential routes to the destination, it also ensures that at least one viable path is available.

Furthermore, our game design encourages players to uncover shortcuts and alternative routes to the endpoint, introducing an element of strategic decision making, as illustrated in Figure 7.5 (initial map). Notably, the map featured in Figure 7.5 serves as the initial state of the game and may not be playable immediately. Nevertheless, our approach, which incorporates a fitness function and an algorithm, facilitates the transformation of this initial map into a playable and challenging environment, guaranteeing the existence of at least one navigable route from Point A to Point B.

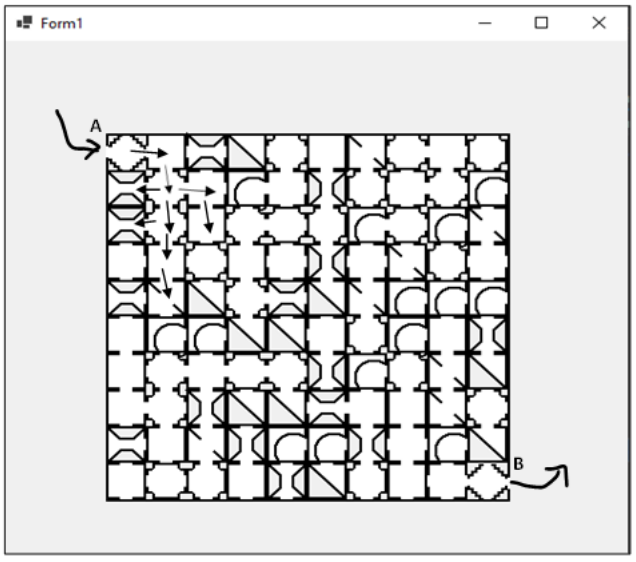


Figure 7.5: The track that the player may take

The concept of a fitness function is widely utilized in the context of op-

timization and decision-making processes. It serves as a mathematical tool designed to measure the quality of solutions. In this study, we formulated an evaluation function designed to appraise the quality and fitness of the map tracks. It is important to note that our assessment relied on the existence and condition of the node's doors.

For example, as illustrated in Figure 7.6 and 7.7, each piece within the generated map represents a node, and these nodes are interconnected through links. Each node maintains two parents (previous nodes) and two children (next nodes). This transformation from a map to a node-link system is visually represented in Figure 7.7, where each node corresponds to a square in the original map and the connections between nodes symbolize the doors. A detailed explanation of the functionality of these doors is provided in our previous discussion in Figure 7.4.

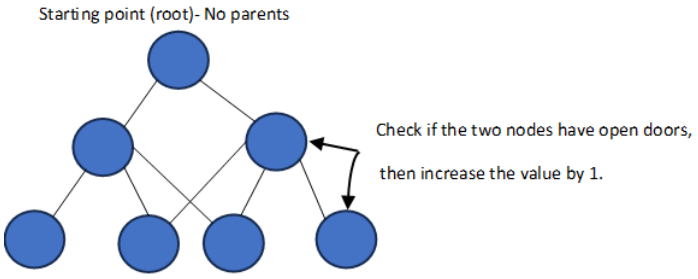


Figure 7.6: represent the map using nodes

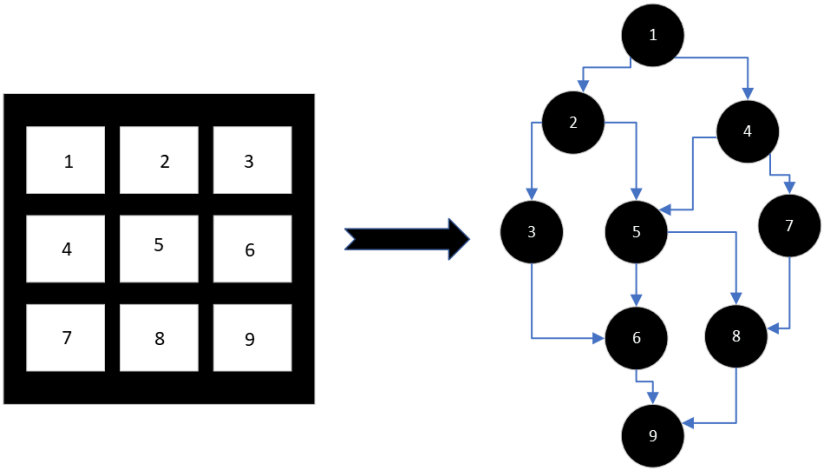


Figure 7.7: Example of transformation a map to graph of nodes

To calculate the fitness value of the entire map, we assessed the connectivity of the nodes and their children using an iterative process, counting open doors until the end node was reached. The fitness value was determined by the openness of the doors along each link, with each open door contributing one unit to the fitness score, reflecting an enhancement in the map's overall fitness. Additionally, we integrated the number of doors opened from the endpoint into the fitness function. This inclusion considers the possibility that initially closed doors may contribute to the improvement in map fitness, particularly when there is an open track from the end.

The fitness function emphasizes the number of doors along the shortest path from start to finish, effectively assessing the route difficulty. A penalty is applied if the route fails to reach the exit node, and the severity of the penalty is multiplicative: a factor of 1.0 for successful completion, 0.5 for halfway progress, and 0.25 for a quarter of the way. To ensure an equitable distribution and prevent dominance by any tile type of map, an additional function assessed the overall map quality. This function further refines the map fitness evaluation by incorporating diverse quality factors of 1, 0.7, 0.5, and 0.25

based on the number of tiles in the map. These factors are multiplied by the overall fitness function, strategically weighting the importance of each map-quality aspect. This adaptive approach enables varied fitness assessments, accommodating diverse scenarios and prioritizing specific map characteristics as required.

This comprehensive approach concludes with the development of the following algorithm.

Algorithm: Calculate Fitness Value

Input: List of Graph Nodes

Output: Fitness value

1. **Initialization: Set the initial values.**
2. **Calculate Track: Count the number of open doors from the starting Point.**
3. **Calculate Trace: Count the number of open doors from the ending point.**
4. **Calculate Penalty Scores (P1 and P2): Evaluate the penalty scores for both Track and Trace.**
5. **Calculate Difficulty Score: Assess the difficulty of the route.**
6. **Calculate Map Quality: Assign a value of 1 if different tiles are fairly distributed.**
7. **Calculate Fitness Value: Use the formula :-**
   ((Track × P1) + (Trace × P2)) × Difficulty Score × Map Quality.

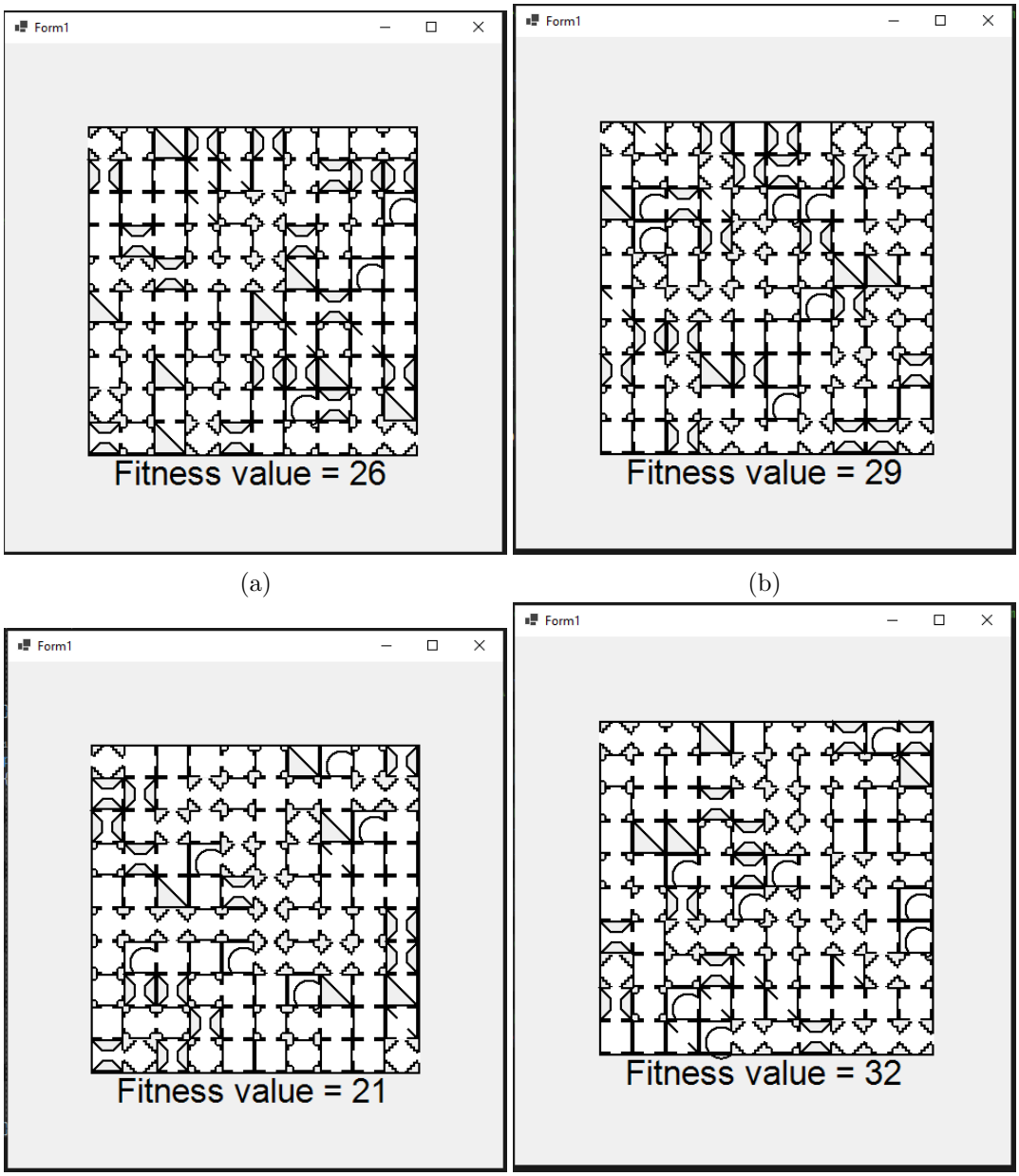
   Return Fitness value

End of Algorithm

7.3.4 Metaheuristics Implementation

The initial stage of the study implemented three meta-heuristic approaches (GA, ABC, and PSO) and utilized these algorithms to generate game maps to gain knowledge

of their relative performance in producing game levels. Here, the performance is determined in two dimensions, namely, the effectiveness of the algorithms in terms of their ability to converge to a better solution in a fixed number of iterations, and the consistency of this effectiveness across multiple attempts to generate maps. Figure 7.8 shows some typical maps generated in this study, noting that the track between points a and b should be open and have many challenges of direction in each would be desirable in a map that was intended to be playable. However, these challenges may not directly affect algorithm performance. The focus of this study was to evaluate and compare the performance of different algorithms, likely in terms of efficiency or effectiveness, rather than specifically addressing the playability of maps. Future studies could address the playability aspect by developing an improved fitness function.



(a) (b)

(c) (d)

Figure 7.8: Different maps generated randomly

7.3.4.1 Genetic Algorithm (GA) Implementation

The GA serves as the main approach employed in this investigation and is included as a foundational benchmark for assessing other methodologies. It stands out as the

most widely utilized evolutionary algorithm, drawing inspiration from the fundamental principles of natural selection and the survival of the fittest. The GA framework systematically iterates through a pool of potential solutions. Within each generational cycle, individuals are chosen from the current pool to act as parents, contributing to the offspring generation for the subsequent iteration.

The selection of parent individuals heavily favors those possessing higher fitness levels within the population, aligning with the core tenet of survival of the fittest. Offspring generation is accomplished by the application of genetic operators, encompassing both crossover and mutation mechanisms [122] [123]. Across successive generations, the population undergoes an evolutionary process, progressively converging towards an optimal solution. GAs have been extensively applied across a broad spectrum of domains including operations research, engineering, and scientific endeavors [125] [146] [123] [157] [64].

Notably, GAs have garnered considerable recognition within the realm of computer games, particularly as a cornerstone of SBPCG. Their utilization spans a multitude of facets within the gaming landscape, encompassing the generation of game contexts for massively multiplayer online games [131], the creation of levels for platform games [76], the design of content for tower defense games [132], quest generation [87], and the crafting of intricate game levels in the domain of dungeon crawler games [80], among various other applications. Our study primarily employs GAs as the chosen meta-heuristic method, following these procedures once the initial population is generated.

1. **Selection of the Parents**

   The first step involves selecting a set of two maps from the population (referred to as parents). We used three different strategies: a biased roulette wheel, tournament, and a combination of both. The biased roulette wheel selection strategy is a variation of the roulette wheel selection technique. In this strategy, the wheel is rotated around a fixed point on the perimeter. The region of the wheel that reaches the fixed point first is chosen as the parent. The same

process is then repeated to select the second parent. This approach ensures that individuals with a larger wheel portion have a higher probability of being selected as parents. Physically, this means that individuals occupying a larger section of the wheel are more likely to align with a fixed point as the wheel rotates. Consequently, the likelihood of a specific individual being selected is directly proportional to the size of the corresponding pie on the wheel. In other words, individuals with higher fitness or probability values are more likely to be selected as parents in this selection strategy.

In the implementation, the following steps are applied:

(a) Calculated the Total Fitness

(b) Calculated the proportion of each individual Pi according to the following equation 7.1:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{7.1}$$

Where N is the number of maps in our population and fi is the fitness value of each map

(c) Calculate the cumulative proportion CP_j of each map according to the following equation 7.2:

$$CP_j = CP_{j-1} + P_j \tag{7.2}$$

(d) Generate a random number R between 0.0 and 1.0

(e) Starting from the top of the population, the individual for which CP_j goes above R is the selected map (parent).

(f) The same process must be done in selecting the other parent.

The second selection strategy used in our approach, GA, was the tournament method. Tournament selection is a widely used method to identify the most suitable individuals from the current generation. In this process, we created a group consisting of K individuals and conducted a tournament using the K-way tournament selection approach. In our attempt, K was set to two, and the mem-

bers of this group were randomly selected. Among the selected individuals, we selected the fittest candidate and promoted it to the next generation.

This tournament selection process was repeated multiple times, allowing us to gradually identify the most optimal candidates for the succeeding generation. Despite the previous selection methods, we incorporated a hybrid strategy that combined biased roulette-wheel selection and tournament selection as our third approach. During each iteration, we generated random numbers between 0 and 100 to determine the selection method. If the generated number is less than 50, we utilize the biased roulette-wheel method; otherwise, we use 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 map. The map with the highest weight was more likely to be considered for further processing and potential improvement.

3. **The Reproduction step**

   Moving on to the third step, we have the reproduction process, in which the maps 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 maps are swapped to generate offspring. This step facilitated the exploration of different genetic combinations, as shown in Fig. 7.9.

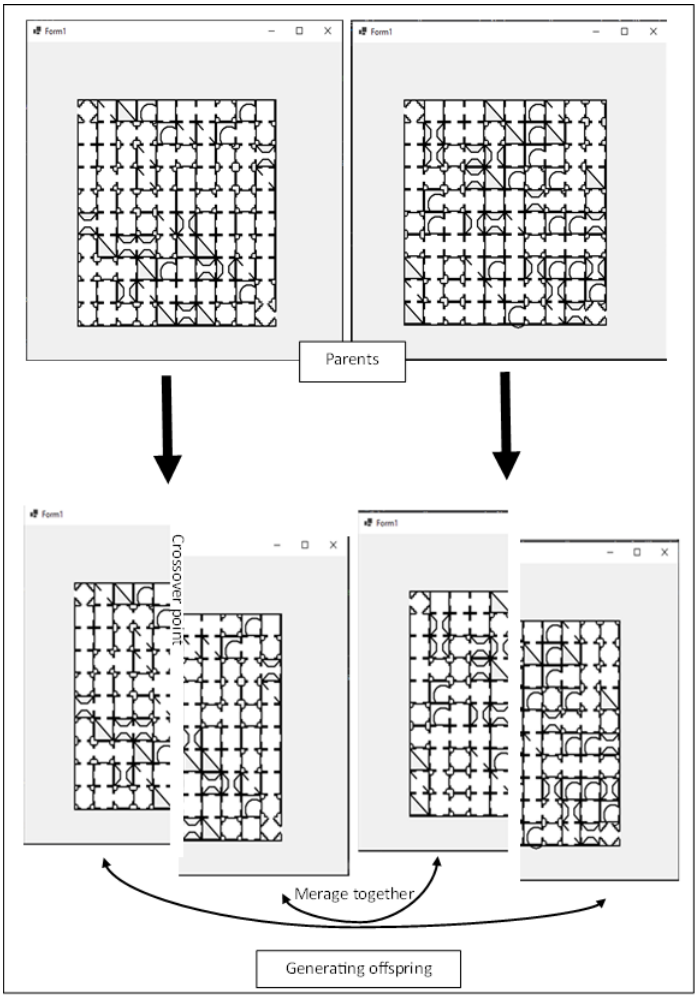


Figure 7.9: Generating Offspring

4. **Mutation Process**

   Mutation serves as the final step in the GA algorithm. In this crucial stage, specified genes (referred to as parts of the map) within the newly formed offspring transform. These genes were selected at random for modification, enabling the exploration of new values and potential enhancements to the genetic diversity of the population. The assigned mutation rate of 0.05 indicates the proportion of genes that are subject to transformation. By iteratively performing these steps, the GA aims to optimize and evolve the population, leading to improved solutions over time. These steps provided an overview of the GA process used in our research methodology. The actual implementation on the CSharp plat-

form involves additional details and specific parameters based on the research objectives and problem domain, as shown in Figure7.10.

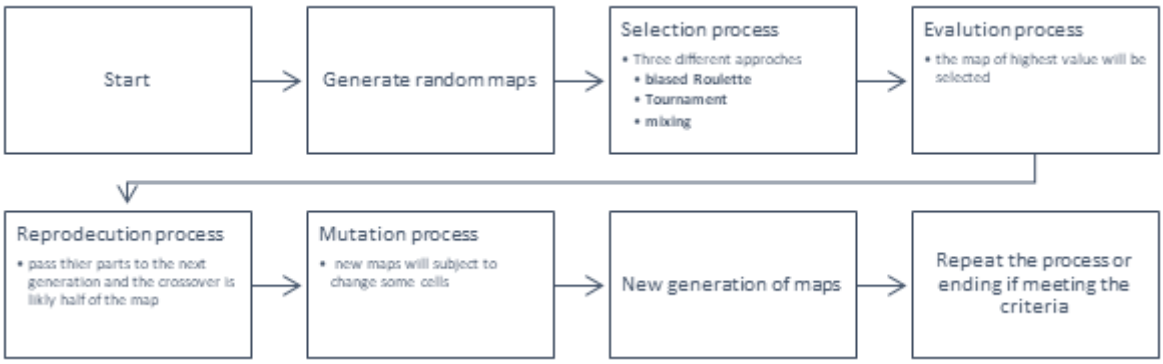


Figure 7.10: Steps of GA approach

7.3.4.2 Artificial Bee Colony (ABC) Implementation

The ABC algorithm represents the second metaheuristic approach explored in this study, drawing inspiration from animal behavior and aligning itself with the realm of swarm intelligence (SI). SI encompasses techniques that rely on a collective of workers, particles, or agents collaborating to seek an optimal or near-optimal solution to a given problem. In the context of the ABC algorithm, this collective behavior mimics the actions of simulated honeybees as they forage for food sources [37]. The core principles underlying swarm intelligence are self-organization and division of labor, both of which suffice to engender swarm-intelligent behavior [133] [147]. These principles resonate strongly within natural honeybee colonies, and are complemented by the incorporation of satisfaction principles.

Although the popularity of the ABC algorithm may not rival that of Genetic Algorithms (GAs), it remains a widely adopted technique that has proven effective in tackling diverse optimization problems. For instance, it has been instrumental in enhancing the speed and precision of gray image segmentation [158], particularly in refining the search process for neighboring edge points to obtain the final edge detection outcome. Furthermore, ABC has found utility in addressing antenna array design challenges, demonstrating its advantages through comparative evaluations

with the four other algorithms. The presentation and analysis of the results affirms its suitability for addressing antenna design complexities [68]. In practical engineering optimization, ABC has been harnessed as an efficient means of addressing interval optimization problems by introducing an innovative approach that leverages ABC to bolster interval credibility [134]. However, the ABC algorithm is the second approach that we used to generate an optimized map through a three-phase approach as follows:

1. **Initial Population:**
   In the first phase, 600 random maps are generated. Each map, denoted by X_i, represents the position of the food source. These maps served as the initial solutions for the optimization process.

2. **Employee Bee Phase:**
   During the second phase, the employee bees actively engage in updating the maps. They evaluated the quality (fitness) of their assigned solutions, and explored the search space within the population. By employing local search operators, such as mutations, the bees update their solutions. In this case, the mutation rate of 0.05 is set, indicating the proportion of the map that is submitted to mutation during each update. This phase contributes to the ability of bees to adapt and refine their solutions within an evolving landscape.

3. **Onlooker Bee Phase:**
   In this stage, onlooker bees observe the solutions of employed bees and their corresponding fitness values. Onlooker bees select the best food sources (maps) based on the higher likelihood of these sources being in the best positions. They evaluated the maps and selected those with superior fitness values. This phase emphasizes the exploration of promising areas within the search space. To accomplish this, onlooker bees considered the probability P(χi) of the best direction, as defined in Equation 7.3. By leveraging this probability, they aim to discover new solutions (maps) that offer potential improvements. The primary objective of this phase is to enhance the quality of the

maps through iterative exploration of superior solutions.

$$P(_i) = \frac{f_i}{Maxf} \tag{7.3}$$

where f_i is the fitness value and $Maxf$ is the maximum fitness in the population.

4. **Scout Bee Phase:**

   In the ABC method, the scout bee phase acts as the final stage. When an employed bee exhausts its search for a particular solution without improvement, it transforms into a scout bee. Scout bees ignore their current solutions and embark on generating new solutions at random, thereby opening new possibilities for exploration. This process of exploring unknown locations enables the algorithm to expand its search limit and uncover potentially superior solutions. By embracing novelty and exploring unexplored areas, scout bees contribute to a continuous search to optimize the solution space.

By iterating through these three phases, the ABC algorithm strives to find an optimized map that represents the best solution to the problem at hand. It combines exploration and exploitation strategies to efficiently search for the solution space and improve the quality of the generated maps. Figure 7.11 illustrates this process.

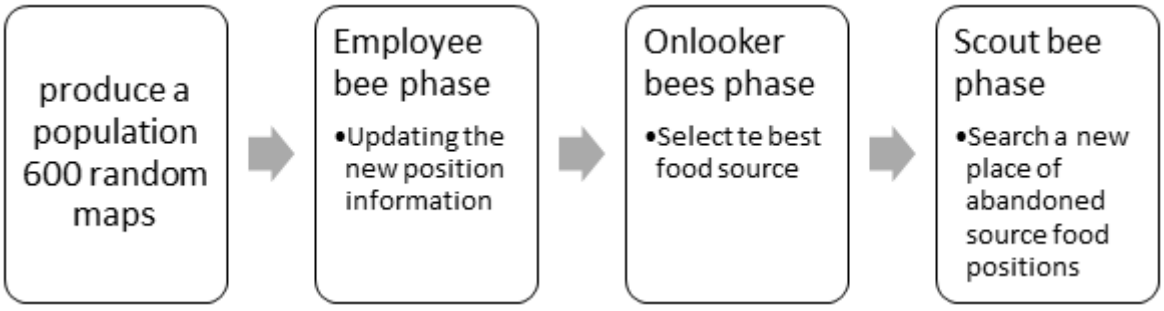


Figure 7.11: Steps of ABC process

7.3.4.3 Particle Swarm Optimization (PSO) Implementation

PSO, such as the ABC algorithm, belongs to the swarm intelligence 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 best position, and global best position accomplished by any particle in the search space. This iterative process continues until the best global position is reached [159].

PSO has been applied to a wide range of problems including robot path planning [137], ship design [138], and a limited number of applications related to video games [86]. There are a limited number of applications of PSO as an algorithm for PCG tasks. For instance, the use of PSO to create content for an endless platformer game has successfully demonstrated its appropriateness and underlined its clear advantages over GAs in terms of both effectiveness and efficiency. [87].

PSO was the last method discussed in this study. In contrast to GA, PSO does not have any modification operators, such as Crossover or Mutation. Instead, PSO considers particles as prospective solutions that move through the problem space by following the best position of the current best particles. PSO has been shown to require fewer computations to solve a problem than GAs [148] and is considered to be more efficient in terms of both memory and speed [149]. PSO inspired the development of an optimization method to solve complicated mathematical problems based on the behavior of a flock. It is difficult for a swarm of birds to collectively determine a landing site when flying over a given area. This depends on various factors, including optimizing the food supply while minimizing the risk of predators. In this sense, the birds' movement may be regarded as choreography; the birds fly in lockstep for a period until they indicate the ideal landing place and the entire flock lands simultaneously [149]. The classical version of PSO is based on determining a variable represented by vector X, as shown in Equation 7.4:

$$X = [x1, x2, x3.xn] \tag{7.4}$$

Here, vector X represents the initial population of the maps, which is 600. This

implies that n= 600. Depending on the suggested evaluation method of function f(X), X should be maximized. X is a position vector, and n is the dimension of the vector as well as the number of variables that the problem may describe. f(X) is also known as the fitness function and can be used to determine how good or bad a location of X is, based on numerous survival criteria[160]. In a swarm of P particles, there exists a position vector, as shown in Equation 7.5, and a population velocity vector, as shown in Equation 7.6, where T represents the number of iterations.

$$X_i^t = (x_{i1}x_{i2}x_{i3}..x_{in})^T \tag{7.5}$$

$$V_i^t = (v_{i1}v_{i2}v_{i3}..v_{in})^T \tag{7.6}$$

These vectors are updated through dimension j according to Equations 7.7 and 7.8, where i=1,2,3..., P and j= 1,2,3...n. ω represents the weight factor.

$$V_{ij}^{t+1} = (V)_{ij}^t + c_1 r_1^t (pbest_{ij} - X_{ij}^t) + c_2 r_2^t (gbest_j - X_{ij}^t) \tag{7.7}$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \tag{7.8}$$

Implementing PSO to generate the best map requires a project to consider the specific settings. Therefore, the network topology method was used to update the map position. It also regularly updates its weight factor ω, which depends on map convergence to the global position of the swarm. However, the weight factor ω is adjusted when seeking new maps, if the map does not improve. Acceleration coefficients C1 and C2 maintain a balance between the exploration and exploitation strategies. C1 determines the percentage of information retained by the map, whereas C2 defines the information obtained from the global position.

7.4 Experiments Analysis

The experiments conducted in this study were implemented using the CSharp Windows Form applications (Visual Studio 2022). The experiments were performed on a computer system equipped with an Intel® Core™ i5-9500 CPU running at 3.00GHz with 16 GB of RAM. The objective of these experiments was to compare the performance of three different meta-heuristic search algorithms in the context of procedural content generation, specifically for map generation. The results presented herein were aimed at evaluating the efficiency and effectiveness of each selected metaheuristic method. To ensure a fair comparison, the maximum iteration number was set to 200, and the population size was set to 600 for all methods. However, a lack of convergence criteria was noted, as in many real-world applications, there may be benefits in running the algorithms until convergence is achieved.

Furthermore, each method and its respective settings were executed ten times. This approach allowed for a more robust evaluation, and the average of the outputs from these ten runs was considered for the study. In addition to the average values, statistical information, such as the best values, worst values, and standard deviations, was calculated and analyzed. This comprehensive statistical analysis provides a deeper understanding of the performance and variability of different metaheuristic algorithms.

The CSharp programming language and specified hardware configuration were chosen to ensure the reliable and consistent execution of the experiments. The results obtained from these experiments contribute to a better understanding of the capabilities and performance characteristics of evaluated meta-heuristic search algorithms in the domain of procedural map generation.

The subsequent subsections (7.4.1, 7.4.2, and 7.4.3) provide detailed insights into the application of each method, and explain their specific implementation details and characteristics. Finally, Section 7.4.4 will present a comparison of the algorithms, enabling a comprehensive evaluation and highlighting their relative strengths and weaknesses.

7.4.1 Genetic Algorithm Application Results

The application of GA is discussed in Section 7.3.4.1, and the results presented here relate to the performance of the three different selection approaches that have been implemented. Figure 7.12 shows the improvement in the fitness function over 200 generations for which the algorithm was run using the hybired selection approach (mix). Here, the line indicates the average fitness for each generation across ten runs of the algorithm. The error bars indicate the average variation in the individual runs.

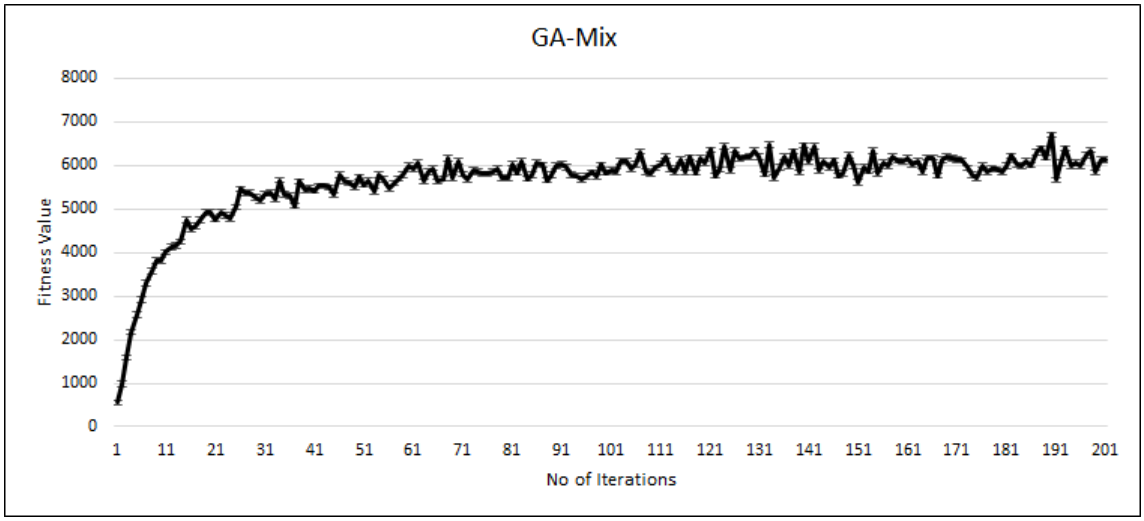


Figure 7.12: GA-Mix

The algorithm exhibits a diverse convergence pattern that is commonly associated with evolutionary approaches. Initially, there was a relatively sharp convergence within the first 15 generations, reaching 4756.3 points. Following this, there was a gradual and consistent increase from 15 to 70 generations. Beyond this point, the rate of improvement slows down, ultimately peaking with an average fitness value of approximately 6705 points. Notably, minor variation was observed, with an average of approximately 6000 across individual runs, underscoring the stability and consistency of the results. However, the curve demonstrated instability with small fluctuations in values between generations 20 and 200, where there was only a minimal improvement within the range of 5000–7000 points. The reasons for this phenomenon may be attributed to constraints in the fitness function, such as the requirement for

an available track and a fair distribution of tiles. Figure 7.13 shows visual samples of the maps generated using this approach.

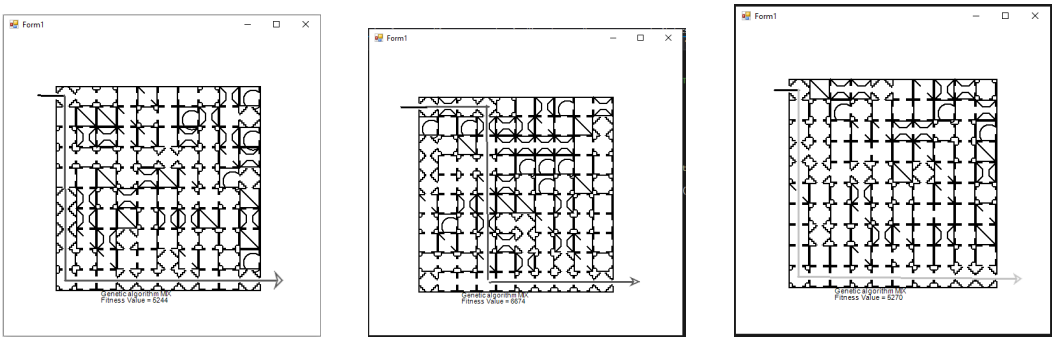


Figure 7.13: Samples of Map generation using GA-Mix Approach

The results obtained using GA with a biased selection method are presented in Figure 7.14. Although the initial average fitness level is similar to that depicted in Figure 7.12, an observable trend indicates marginally faster initial convergence. The algorithm achieved a fitness value of 4650 points within the first 11 generations. After that, there is a more significant increase in values until generation 60, followed by a period of fluctuation with some peaks and valleys. From stage 60 to around 160generation 160, the curve shows a generally increasing trend, although with some fluctuations. Beyond that, there seems to be a relatively stable phase, with values oscillating around a certain range. In summary, the curve starts with fast growth, experiences a more pronounced increase, undergoes fluctuations, and eventually reaches a relatively stable phase. These fluctuations might indicate periods of variability or influence from different factors at various points along the curve averaging approximately 5780.3 points. Upon comparing the variability across runs, it is noted that the biased selection method demonstrates a slightly reduced degree of variability in contrast to the mixed selection method. Specifically, the range is between 230 and 631, whereas the result in Figure 7.12 covers between 117 and 807.

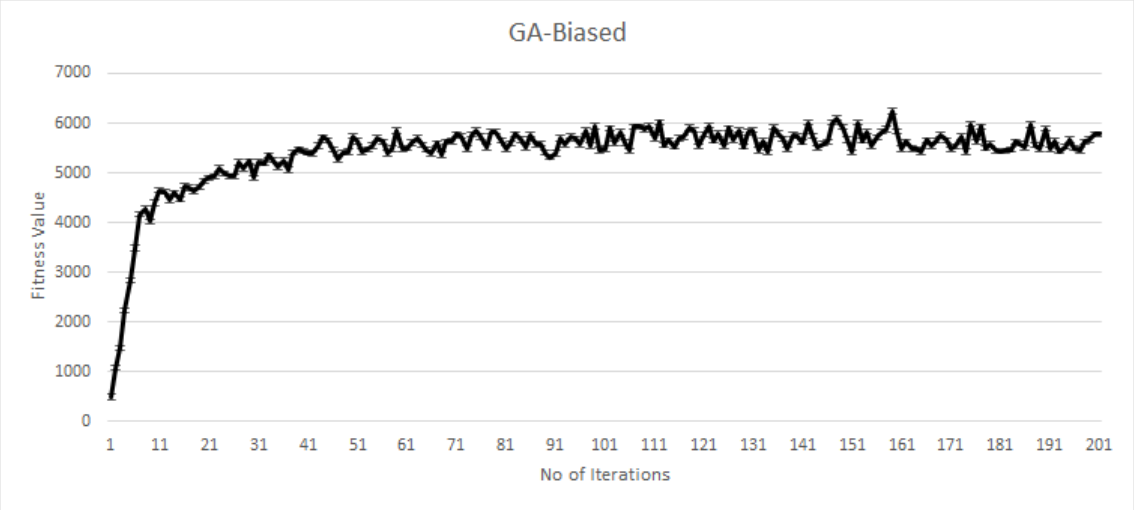


Figure 7.14: GA-Biased

Continuing the exploration of the GA biased selection approach, Figure 7.15 provides visual representations of maps generated using this particular method.

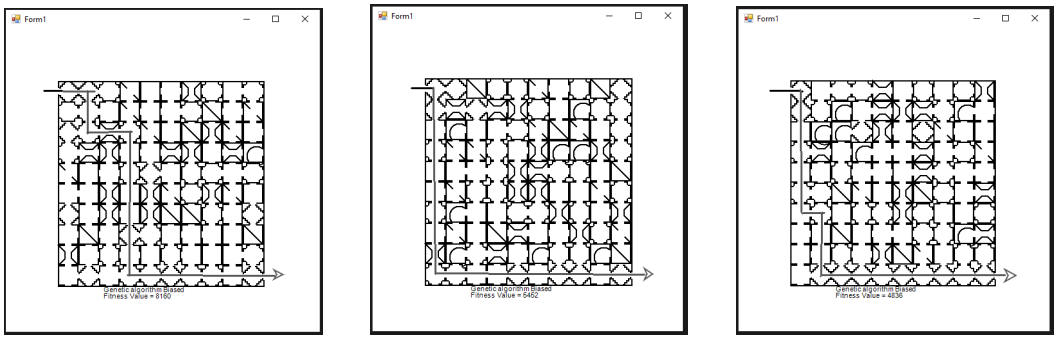


Figure 7.15: Samples of Map generation using GA-Biased Approach

Figure 7.16 shows the results for using the tournament selection approach. During the initial 30 generations, there was an obvious and sharp rise in fitness values, indicating a significant early stage improvement in the solutions. Subsequently, from generations 30 to 90, the curve experienced periods of fluctuation and alternation, indicating a dynamic exploration of different solutions. Around generation 60, a plateau is reached, maintaining a stable fitness level until approximately 90 generations.

The route shifts with a second phase of notable growth starting around Gen-

eration 90. Fitness values steadily increase, indicating continuous enhancements in solutions generated by the GA Tournament. As the curve progresses towards the end, it stabilizes, and minor fluctuations in the fitness values become apparent. This stabilization marks a potential convergence to a relatively optimal solution, reflecting a slowdown in the overall fitness improvement. The observed fluctuations in the middle phases may be attributed to the exploration of diverse solution spaces by the algorithm.

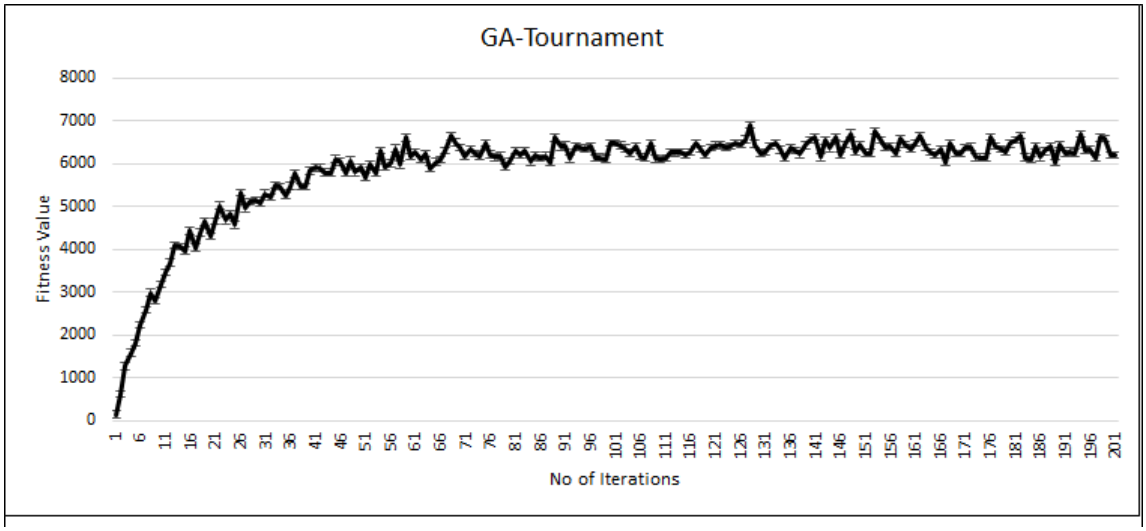


Figure 7.16: GA-Tournament

Figure 7.17 presents visual representations of maps generated through the GA Tournament process.

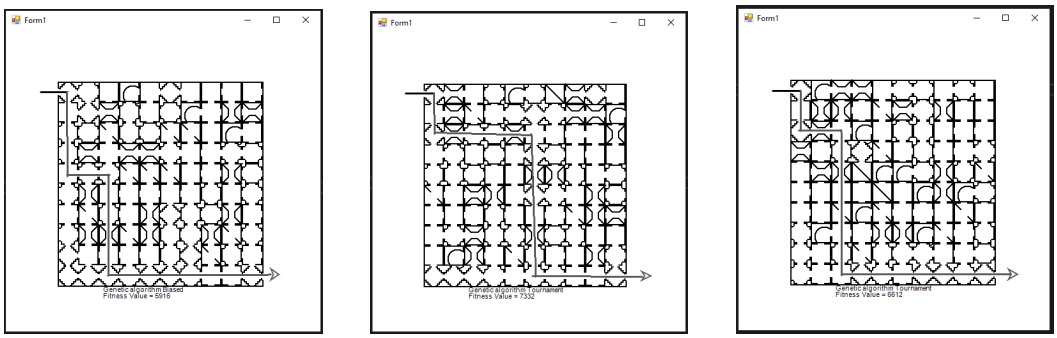


Figure 7.17: Samples of Map generation using GA-Tournament Approach

7.4.2 Artificial Bee Colony Application Results

Figure 7.18 shows the results of multiple runs of the ABC algorithm, which show a marked difference from the GA results. The initial speed of convergence is much slower, and while 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 the GA convergence. The initial average fitness value was marginally not significantly different from the results for the different GA implementations, and the convergence was slower, reaching the previously indicated threshold value of 4000 in just over 150 iterations. However, unlike the GA implementations, the convergence is asymptotic, and the improvement continues over the entire 200 iterations, leading to a final average fitness value of 4512 points. small fluctuations between generations 40 and 110. Notably, compared to GA, ABC has a lower performance, displaying an important difference in outputs.

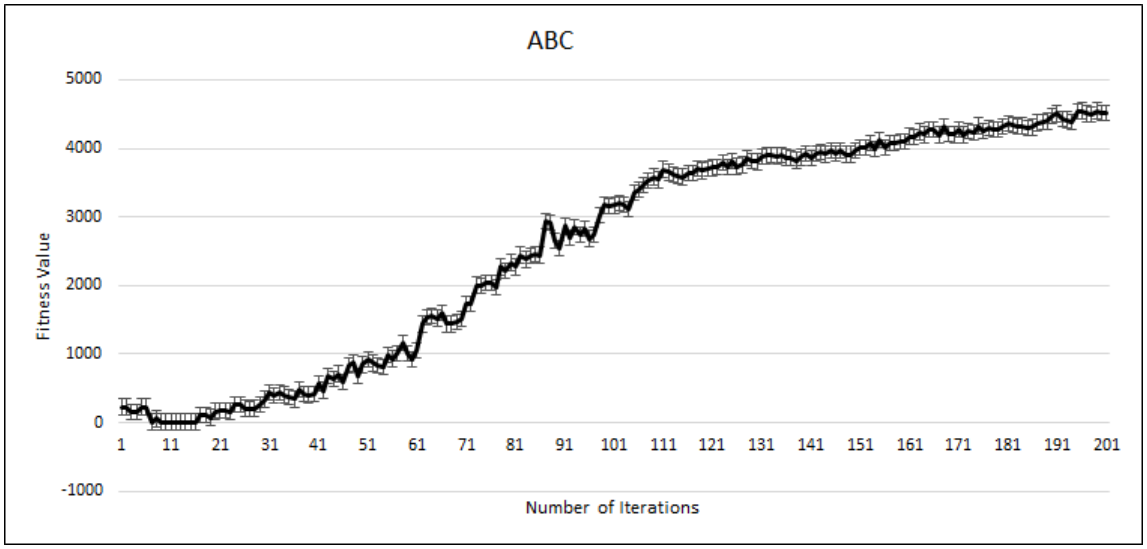


Figure 7.18: ABC results

Figure 7.19 displays some map samples related ABC

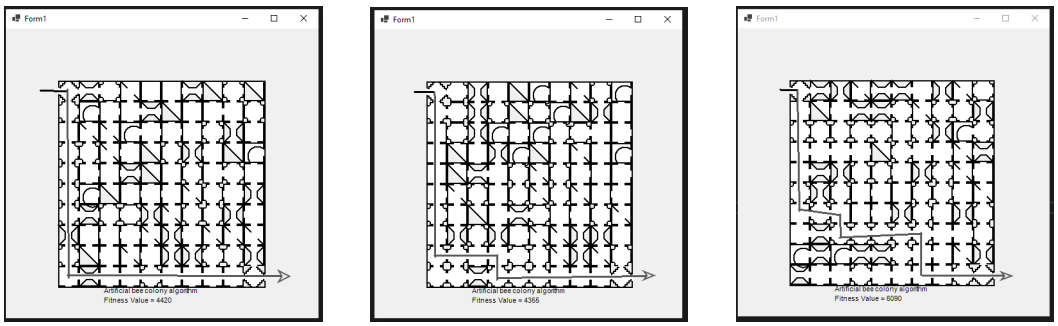


Figure 7.19: Samples of Map generation using ABC Approach

7.4.3 Particle Swarm Optimization Application Results

In alignment with our genetic algorithm (GA) implementation, we implemented two variations of the particle swarm optimization (PSO) algorithms using an approach similar to that in our previous research [151]. These variations involve different values of C2 acceleration coefficient in Equation 7.7, which determines the amount of information extracted from the global position during the search. In both implementations, C1 was assigned the value of 2. The first implementation used a C2 value of four, whereas the second implementation used a C2 value of five.

The results of the first implementation are shown in Figure. 7.20. Compared to the results obtained from both the GA and ABC algorithms, the PSO algorithm demonstrates a fast initial convergence that reaches the threshold fitness value of 2002 points after 52 iterations. The convergence curve then becomes almost asymptotic, indicating a gradual and consistent improvement over the remaining iterations. The final average fitness achieved in fewer than 200 iterations was approximately 2799.87. Overall, the final fitness quality and variability around the average were less than the best results obtained using the GA and ABC. However, the samples of maps produced by using PSO are depicted in the figure 7.21

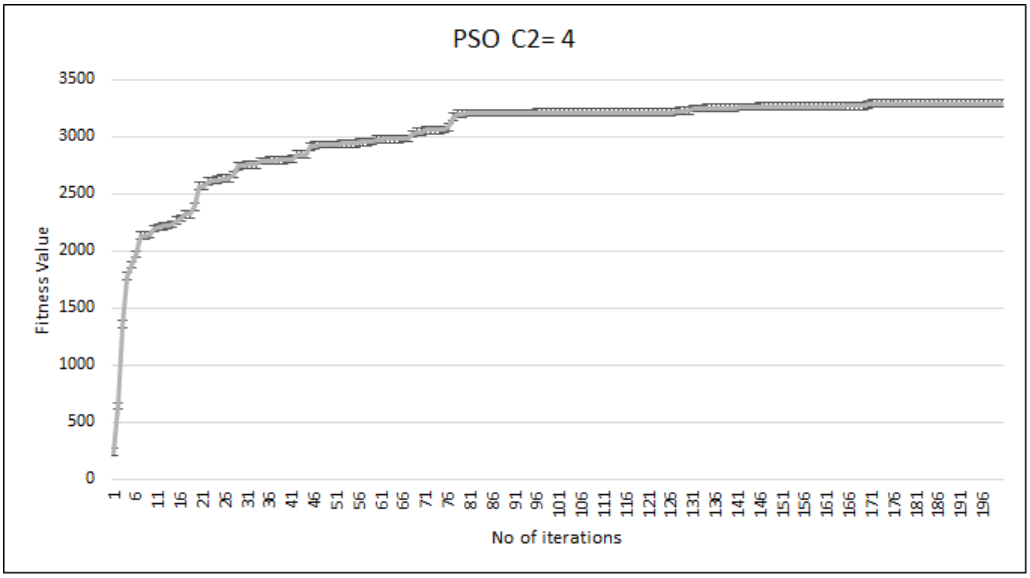


Figure 7.20: PSO#1

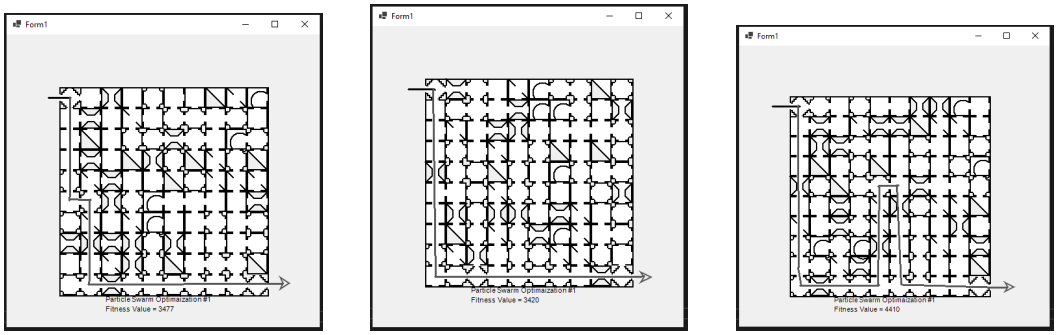


Figure 7.21: Samples of Map generation using PSO Approach where C2=4

The results of the second implementation of the PSO algorithm, which places greater emphasis on global position information, are shown in Figure 7.22. These results exhibit minimal variation in the convergence behaviour, with the threshold value of 2026 points being reached in just over 56 iterations. There was only a slight improvement in the final average fitness function value and variability, approximately 2716.93. It is important to mention that these results have no major difference from the first implementation. In addition, Figure 7.23 illustrates additional sample maps.

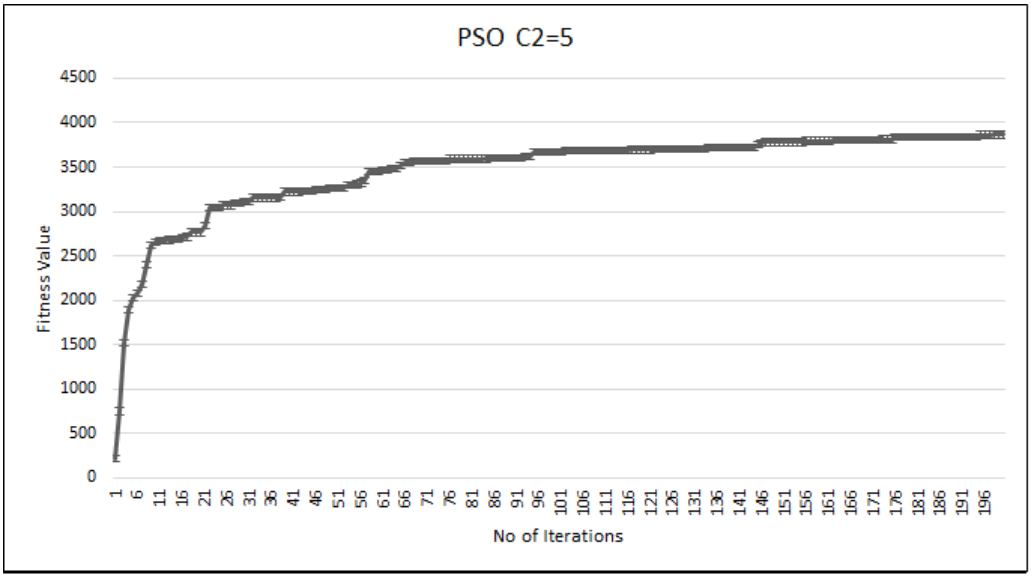


Figure 7.22: PSO#2

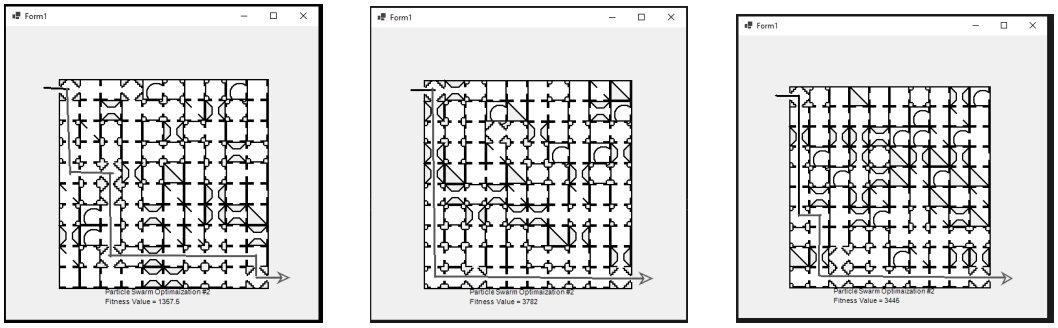


Figure 7.23: Samples of Map generation using PSO Approach where C2=5

7.4.4 Comparison Analysis

The results presented in Figures 7.12–7.22 provide a visual representation of the algorithm performance and offer an initial assessment of their relative merits. Table 7.1 complements these findings by providing additional information on the quality of solutions obtained by each algorithm across multiple runs. Specifically, these data correspond to the variation observed in each figure at the 200-iteration mark.

Table 7.1 compares the performance metrics of various optimization algorithms, namely, GA (Mix), GA (Biased), GA (Tournament), ABC, PSO#1, and PSO#2. The eval-

uation was based on the worst fitness score, the best fitness score, and the standard deviation.

	GA(Mix)	GA(Biased)	GA(Tournament)	ABC	PSO#1	PSO#2
Average	6128	5780	6207.6	4519.4	2799.87	2716.93
Worst	5236	5040	5720	4095	1128.4	1156.9
Best	7998	7110	7626	5110	3835	3780
SD	807.312	631.17	535.23	315.10	1083.09	997.53

Table 7.1: Comparison of different algorithms

In terms of the worst fitness score, GA(Tournament) achieved the highest value of 5720, surpassing the other algorithms, even in the worst-case scenario. PSO#1 and PSO#2 recorded the lowest fitness scores of 1128.4 and 1156.9, respectively, indicating their ability to avoid highly suboptimal solutions more effectively than GA.

Moving on to the best fitness scores, GA-Mix demonstrates the highest performance with a score of 7998, showcasing its ability to converge towards average optimal solutions. Other versions of the GAs showed the second-highest scores. ABC closely follows the second-highest fitness score of 5110 as an another algorithm, whereas PSO#1 and PSO#2 achieved scores of 3835 and 3780, respectively.

These results suggest that GAs and ABC have greater potential for finding superior solutions than the other algorithms. Analyzing the standard deviation, ABC achieved the lowest value of 315.10, indicating consistent and reliable performance. Conversely, PSO#1 exhibits the highest standard deviation of 1083.09, implying greater variability in the fitness scores. The remaining algorithms, including GAs and PSO#2, displayed standard deviations ranging from 997.53 to 535.23, indicating moderate variability in their performance.

In summary, although GA-Tournament obtained the highest worst fitness score among the algorithms, it also achieved the best fitness score and displayed moderate variability, indicating consistent performance, even in unfavorable scenarios.

Therefore, GA Tournament stands out as a strong contender for optimization tasks, where attaining the best possible outcome is paramount. However, factors such as the computational efficiency and suitability to specific problem domains should also be considered when selecting the most appropriate algorithm.

In addition, Figure 7.24 provides further insights into the comparison of the results, highlighting the strengths and weaknesses of each algorithm. The GA implementations exhibit good convergence and stability, with the tournament selection approach outperforming the mixed and biased selection methods. The ABC algorithm initially showed slower convergence but exhibited significant improvement after the first 86 iterations, maintaining consistent performance with minimal variability. Conversely, PSO demonstrates the weakest performance, but it is very fast in terms of coverage with a few iterations.

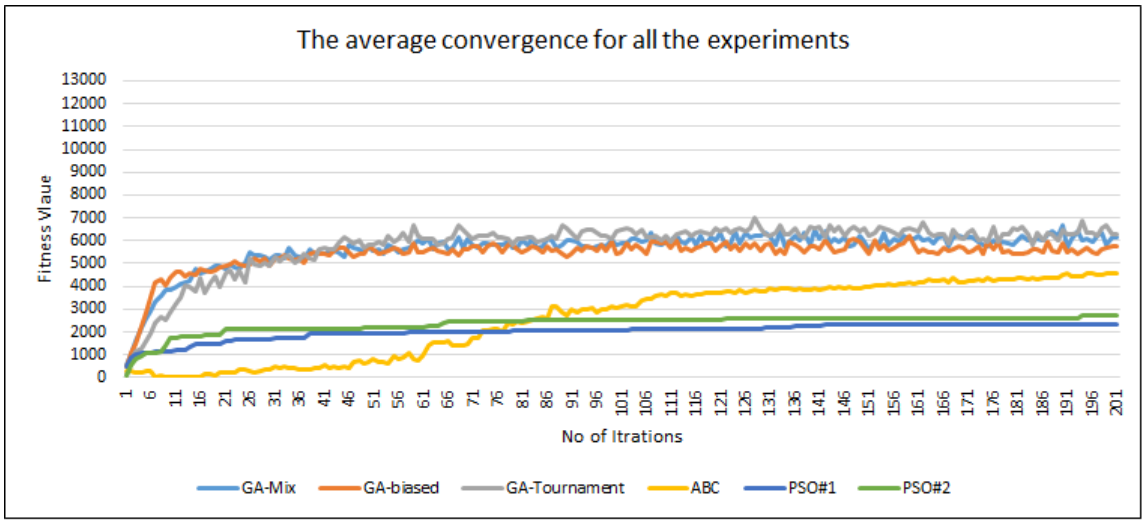


Figure 7.24: The comparison of average convergence for all the algorithms

7.5 Discussion, limitations and Future works

This study investigated the performance of various optimization algorithms in the context of game map generation. Notably, GAs, particularly GA tournaments, have demonstrated remarkable effectiveness, as measured by metrics such as average fitness, best fitness score, worst fitness score, and standard deviation. However, a

critical observation has emerged, as the effectiveness of these algorithms appears to be highly contingent upon a specific task. This is evidenced by our prior research [151], where the PSO and ABC algorithms were identified as the top performers for race track generation. This discrepancy underscores the paramount importance of understanding how the inherent characteristics of a task influence the performance of optimization algorithms.

One possible explanation for this difference could be the settings of the parameters used. Although we kept the settings consistent with our previous research for reliability, slight variations in these settings could have affected the algorithm's performance in map generation. In addition, the metrics we used may not fully capture the nuances of the tasks. For instance, in map generation, achieving a balanced map layout may be more important than achieving the best fitness score, owing to constraints on the number of tiles allowed. Conversely, in race track generation, the focus may be on creating challenging paths with different shapes. Consequently, the fitness calculations required for each task could differ.

Differences in evaluation methods and game design between this study and a previous study [151] may also influence algorithm performance. For example, introducing penalty scores in map generation (to penalize maps with good fitness but undesirable qualities) may explain why PSO did not perform as well as the scenarios without penalties. A similar rationale can be applied to ABC. This observation underscores the multifaceted nature of algorithm effectiveness, which is demonstrably influenced not only by the inherent characteristics of the task but also by the specific design choices and evaluation metrics employed.

Although all three algorithms (GA, ABC, and PSO) can perform both global and local search, their primary mechanisms and emphasis on these aspects differ. GAs influence the population dynamics and genetic operators in exploring the search space. ABC utilizes a three-tiered foraging strategy involving employed, onlooker, and scout bees, whereas PSO achieves a balance between exploration and exploitation through a swarm of particles that dynamically adjust their positions. The optimal choice of algorithm in this context depends heavily on the inherent characteristics of the problem domain and the desired balance between exploration and exploitation. Map generation requires a strong emphasis on local search to refine solutions within

the promising regions of the search space. This suggests that adjusting the parameters of ABC and PSO to prefer a local search may lead to an improved performance in map generation. For ABC, this might involve prioritizing the exploitation capabilities of employed bees while potentially reducing the exploration efforts of scout bees. PSO might involve limiting the influence of the global best position on particle movement, thereby encouraging exploitation within promising local areas. Furthermore, fine-tuning the parameters of all three algorithms to emphasize the local search has the potential to further enhance the optimization performance in the context of map generation.

Our future work could involve further analysis and fine-tuning of algorithm parameters. While we utilized default parameter settings, altering parameters such as population size, mutation rate, crossover probability, swarm size, and inertia weight in PSO may yield different results. Fine-tuning these parameters based on specific problem characteristics can potentially enhance the performance of the algorithms and lead to better solutions.

Further investigation is required to gain a more comprehensive understanding of the observed performance variations and identify the most suitable optimization algorithm for specific PCG tasks. Future work could encompass several key areas. First, qualitative analysis of the generated maps and race tracks is crucial for identifying potential differences beyond those captured by the current fitness metrics. Second, incorporating additional task-specific metrics that more accurately reflect the desired qualities of the generated maps would provide a valuable assessment of algorithm effectiveness. By systematically addressing these avenues for future work, we can refine our understanding of how task characteristics influence algorithm behavior, and ultimately make informed decisions when selecting optimization algorithms for diverse PCG applications.

Further investigation is required to gain a more comprehensive understanding of the observed performance variations and identify the most suitable optimization algorithm for specific PCG tasks. Future work could encompass several key areas. First, qualitative analysis of the generated maps and race tracks is crucial for identifying potential differences beyond those captured by the current fitness metrics. Second, incorporating additional task-specific metrics that more accurately reflect

the desired qualities of the generated maps would provide a valuable assessment of algorithm effectiveness. By systematically addressing these avenues for future work, we can refine our understanding of how task characteristics influence algorithm behavior, and ultimately make informed decisions when selecting optimization algorithms for diverse PCG applications.

Finally, conducting comparative studies with state of the art optimization algorithms or incorporating advanced metaheuristic techniques could provide a deeper understanding of the relative strengths and weaknesses of the algorithms examined in this study. This comparative analysis has the potential to identify the most promising algorithms for specific problem domains and offers valuable insights to guide algorithm selection in real world optimization scenarios.

7.6 Conclusion

This study investigated the performance of various metaheuristic algorithms for Procedural Content Generation (PCG), specifically focusing on game map creation. It aimed to challenge the universal reliance on Genetic Algorithms (GAs) by exploring alternative optimization approaches for generating game levels based on our previous research. Through a series of experiments, this study compared the effectiveness and efficiency of GAs, Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) algorithms in generating game maps. Metrics, such as convergence speed and content quality, were used to evaluate the generated maps.

The results indicate that the GA Tournament achieved superior performance compared to ABC and PSO for this specific PCG task. It demonstrated faster convergence and produced higher quality maps based on the defined metrics. These findings support the notion that the optimal metaheuristic algorithm for PCG depends on both the task itself and chosen evaluation methods.

This study contributes to the field of PCG by demonstrating the potential of alternative metaheuristic approaches. Game developers gain access to a wider range of techniques by integrating diverse algorithms into the content-generation process.

This potentially leads to richer PCG experiences, and fosters the creation of innovative and engaging game content.

In conclusion, this study provides initial insights into the performance of various optimization algorithms for PCG. Future research should focus on fine-tuning the algorithm parameters for improved performance, exploring alternative operators and selection methods within the algorithms, testing algorithms on a broader variety of PCG problems, enhancing the computational efficiency of the algorithms, and conducting comparative studies with more advanced optimization techniques.

Addressing these areas will contribute to the ongoing development and refinement of optimization algorithms for solving complex real-world problems, including those encountered in game content generation.

Chapter 8

Discussion

8.1 Overview

This thesis investigated the potential of metaheuristic optimization methods for generating captivating and diverse game content within the realm of procedural content generation (PCG). Guided by an extensive literature review and comprehensive research methodology, this study explored these possibilities through the lens of the following research questions:

- **RQ1: What are the most common and effective meta-heuristic algorithms used for procedural content generation (PCG) in games and how have these algorithms been applied across various domains within PCG?**
- **RQ2: What are the main challenges and limitations of using meta-heuristic algorithms for PCG and how can these be addressed in future research?**

New research questions (RQ3, RQ4, RQ5, and RQ6) were formulated based on our finding exciting directions for additional studies, see Figure 8.1.

- **RQ3: How do Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compare in terms of solution quality and convergence speed when applied to the task of race track generation in PCG?**
- **RQ4: In the context of map generation for PCG tasks, how are Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compared in terms of solution quality and convergence speed?**
- **RQ5: How stable are the solution quality and convergence speed of Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) when applied to race track and map applications?**
- **RQ6: How applicable are the findings of this comparative study to a broader range of PCG tasks beyond those specifically examined, such as race track and map generation?**

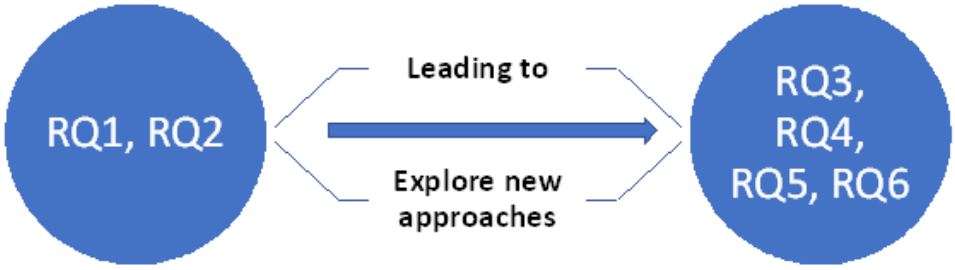


Figure 8.1: Developing Research Questions

These additional questions directly investigate the particular functions and efficacy of three well-known metaheuristic algorithms in the context of PCG: particle swarm optimization (PSO), artificial bee colony (ABC), and genetic algorithms (GAs).

The Genetic Algorithm (GA) is a widely used tool in PCG, as shown by the responses to research question RQ1. We examined possible alternatives and a practical understanding of these algorithms by comparing the performance of GA with that of particle swarm optimization (PSO) and Artificial Bee Colony (ABC) optimization techniques. The rationale behind choosing ABC and PSO is their distinctive yet significant presence in swarm intelligence space [161], [162], [163], [164].

To conduct this comparative analysis, we focused on two representative tasks: race track generation and map generation. By subjecting these distinct scenarios to the optimization capabilities of each algorithm, our objective was to reveal valuable insights that address overarching research questions. Detailed responses to these inquiries are presented in subsequent sections of this chapter.

This study used a quantitative methodology to thoroughly assess and contrast the efficacy and efficiency of the selected algorithms in creating game levels. The convergence speed and solution quality were the two main criteria that were the focus of the investigation. Multiple repetitions of the experiments observing these indicators were performed to ensure reliable outcome. We separated the algorithms into important factors by implementing a strictly regulated experimental design. We maintained methodological integrity and ensured that any performance disparity could be reliably attributed to the fundamental characteristics of each algorithm by excluding other effects.

Although our quantitative analysis using metrics, such as convergence speed and solution quality, established the effectiveness of the algorithms, we recognize the limitations of relying solely on numerical data. Capturing the subjective nuances of player experience, such as ``gaming quality", is complex and multifaceted, encompassing aspects beyond quantification. Nonetheless, understanding the players' experiences is crucial. To bridge this gap, we employed bespoke fitness functions that indirectly capture the key aspects of level playability by evaluating factors, such as structural coherence, diversity of elements, and overall challenge progression. These functions aimed to provide an indirect indicator of player engagement, and modifications were applied uniformly across all the algorithms for a fair comparison.

Expanding on this quantitative foundation, we additionally examined how well the algorithms perform in comparison to the two PCG tasks. The objective of this comparative analysis was to identify the possible patterns, advantages, and disadvantages unique to each algorithm in the context of solving various PCG issues. As we will explore in more detail later in this chapter, by studying the data from both tasks, we can obtain a more thorough knowledge of how each algorithm generalises its capabilities and adjusts to different PCG requirements.

This chapter builds upon the foundations established in Chapters 3, 5, and 7. Chapter 3 utilized a systematic literature review to provide a comprehensive overview

of the current research landscape in Procedural Content Generation (PCG) with a focus on the use of metaheuristics. Chapters 5 and 7 offer additional context and insights, enriching our understanding that this review analyzes the applicability of these findings to the broader field of PCG, providing insights into the prevalence and scope of metaheuristic studies within the literature. Drawing upon these combined analyses, this chapter discusses the use of metaheuristics in the Procedural Content Generation (PCG) field and identifies key trends and challenges in the application of various methodologies for PCG research.

8.2 Prominence of Genetic Algorithms in Procedural Content Generation

Our research, discussed in Chapter 3 [50], shows that metaheuristic algorithms are widely used in procedural content generation (PCG), thereby answering our main research question (RQ1). Among these algorithms, Genetic Algorithms (GAs) are used much more often than others, such as Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO). Approximately 52% of the studies used GAs for creating content, showing how important they are in PCG.

The utilization of search-based and evolutionary algorithms, which share similarities with genetic algorithms (GA), underscores the considerable emphasis in research on evolutionary approaches across various domains. This percentage was 93%.

Chapter 3 [50] highlights the natural affinity between early PCG research and GA/EA algorithms because of their similarity in how designers create game levels. This shared approach to exploration and innovation likely drew researchers toward these algorithms. As GAs and EAs have demonstrated success in level generation, their application has broadened to tackle a wider range of content creation within PCG.

Several factors solidify the widespread adoption of GAs among PCG. Their inherent ability to explore diverse solution spaces, handle complex problems, and adapt to various problem representations makes them well-suited for generating different types of game content. This aligns with previous studies that highlighted the effectiveness of GAs in tasks such as level design and item creation [9], [35],[165] and [77]. Although GAs have undeniable strengths, it is crucial to acknowledge the potential limitations of relying solely on this approach.

Our comprehensive analysis of the 97 research papers presented in Chapter 3 [50] and addressing RQ1 and RQ2 revealed promising opportunities and notable challenges associated with utilizing metaheuristics in procedural content generation (PCG). Although the established algorithms demonstrate effectiveness, there is significant potential for exploring alternative approaches, tailoring them to specific content types, and developing entirely new methods.

The study emphasizes the importance of fitness functions in guiding the algorithm optimization. Remarkably, only 27% of the studies have focused on developing strong functions. Additionally, only 1% of the studies explored combining multiple algorithms, and user feedback was absent in 76%. These findings suggest promising opportunities to enhance content quality and player engagement. Furthermore, thorough content evaluation requires diverse metrics and the involvement of domain experts. However, these aspects are currently lacking in 52% of studies.

Finally, the analysis identified a lack of research addressing ethical considerations related to ownership and authenticity of PCG-generated content. This necessitates further investigation and establishment of guidelines to ensure that ethical implications are adequately addressed.

In our investigation of the use of metaheuristics for Procedural Content Generation (PCG) in Chapter 3, we align with insights from prior research. For instance, Cardoso (2023) [166] expressed concerns regarding the lack of conclusive evidence regarding the effectiveness of various metaheuristic algorithms across diverse ap-

plication domains. This echoes our findings, emphasizing the necessity of exploring alternative approaches beyond the established algorithms to fully realise PCG's potential.

Similarly, Hussain (2019) [167] highlighted the limitations of relying solely on basic performance metrics. They advocated for more in-depth assessments to comprehensively understand the strengths and weaknesses of the different algorithms. This corresponds to our advocacy for the development of robust fitness functions capable of accurately capturing the desired content properties and effectively guiding optimization.

Moving beyond algorithm design, Zamli (2018) [168] suggested a shift in research focus towards the adaptive hybridization of existing algorithms. This resonates with our finding that combining multiple algorithms holds untapped potential for improving content quality and engagement in PCG. Moreover, Malan (2014) [169] underscored the importance of robust fitness functions, particularly in their ability to accurately predict the algorithm performance for unseen problems. Our analysis echoes this sentiment and highlights the crucial role of well-designed fitness functions in effective content generation.

Furthermore, by emphasising the ethical issues surrounding the ownership and authenticity of PCG-generated information, our research advances into uncharted territory. This issue, which is completely missing from the studies, calls for more research and the creation of responsible standards to ensure that ethical implications are appropriately handled.

Finally, our results not only align with previous studies on PCG and metaheuristics, but also indicate new directions, such as ethical issues that need to be investigated further. We can fully realise the promise of metaheuristics in influencing the PCG's future and promoting the production of interesting and varied content experiences by tackling the gaps in our work and expanding on findings from earlier studies.

8.3 Overview of The Specific Task Results

In Chapters 5 [151] and 7 [170], we investigated the effectiveness of three popular metaheuristic algorithms in the domain of procedural content generation (PCG): Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC).

Specifically, Chapter 5 focuses on applying these algorithms to the task of generating race tracks, while Chapter 7 examines their performance in map layout generation. The results obtained in both chapters offer valuable insights into the strengths and weaknesses of each algorithm when applied to different PCG applications.

These chapters directly address RQ3 and RQ4, which are discussed in detail in the following sections.

8.3.1 The Efficacy of Metaheuristics in Procedural Race Track Generation

Developing dynamically captivating racetracks presents particular challenges in the context of Procedural Content Generation (PCG). In Chapter 5, the works of particle swarm optimization (PSO), artificial bee colony (ABC), and genetic algorithms (GA) are described in detail. They have been utilized to handle the complexity of race track development. With this analysis, we seek to gain insight into the advantages and disadvantages of each algorithm and, ultimately, establish the efficacy of one relative to the other in this particular PCG application.

Our inquiry is directed by Research Question RQ3 (How do Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compare in terms of solution quality and convergence speed when applied to the task of race track generation in PCG? ? To address this crucial inquiry, we meticulously crafted robust quantitative measures, focusing on key performance indicators, such as convergence speed and solution quality.

Our effectively planned experimental setup is essential for assessing the various techniques of each algorithm in a neutral manner. This was accomplished by carefully regulating every shared variable between several algorithms, including the population size and iteration count. By ensuring that irrelevant factors were eliminated, we were able to identify and comprehend the precise elements in each algorithm that were responsible for the observed variations in performance when it resulted in creating an engaging racing generation.

Expanding upon the fundamental research conducted in Chapter 5 [151], which examined the unique performance of every algorithm, we acknowledge that there is no widely recognised convergence criterion for PCG applications. It is necessary to continue running the method until convergence is achieved under real-world circumstances. To counteract this possible source of confusion, every algorithm and the corresponding parameter settings were carefully run ten times, and the average result of each run was used in our study.

Additionally, to offer a comprehensive picture of algorithmic performance, a spectrum of statistical measures was employed, encompassing the best, mean, worst values, and standard deviations. This multifaceted approach ensures robust and accurate evaluation of the capabilities of each algorithm.

Table 8.1 and Figure 8.2 present the range of fitness values across the selected algorithms ordered from lowest to highest. The lowest fitness value indicates an optimal solution. It is evident from the data that PSO#1 achieves the most favorable optimal fitness according to this criterion.

	Best	SD
PSO#1	106.88	38.92
ABC	113.98	8.73
PSO#2	115.86	23.17
GA(Mix)	121.75	52.35
GA(Tournament)	137.74	25.28
GA(Biased)	304.30	46.13

Table 8.1: Performance of the Algorithms in the task of race track

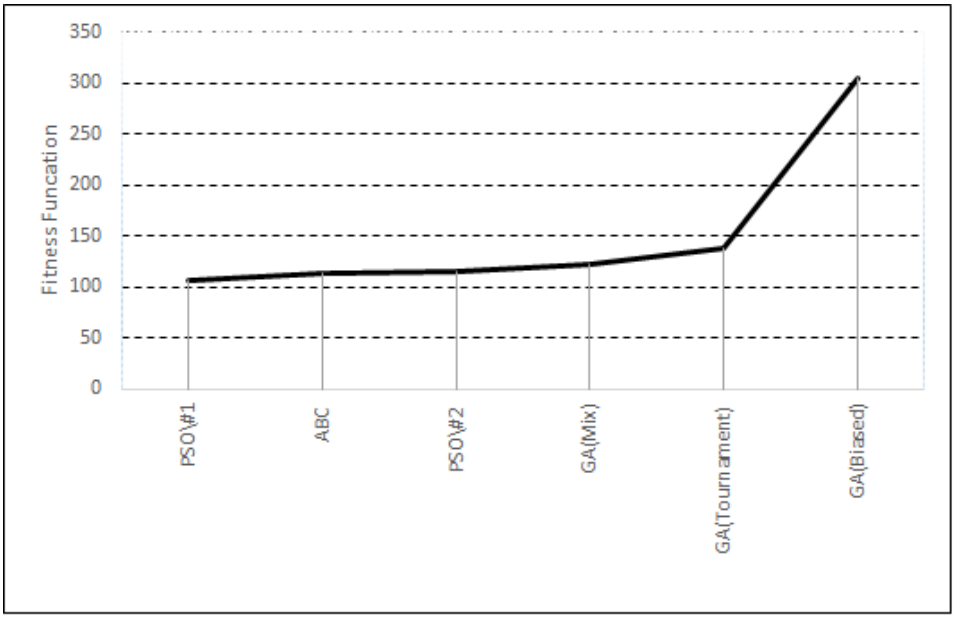


Figure 8.2: Performance of the Algorithms in Race track

From Table 8.1, PSO#1 emerged as a strong contender, achieving the lowest best fitness (106.88) with moderate variability (38.92). This particular variant of PSO indicates its exceptional performance in finding optimal solutions for race track generation. Its competitive performance was further highlighted by a relatively low standard deviation of 38.92, suggesting consistent and stable performance across iterations. PSO#1's ability to efficiently explore the solution space and converge towards optimal solutions makes it a promising choice for procedural content generation tasks.

Furthermore, ABC achieved a commendable best fitness value of 113.98, demonstrating its effectiveness in determining near-optimal solutions. The low standard deviation of 8.73 suggests that ABC maintains a high level of consistency in its performance, with minimal variability across iterations. This stability indicates that ABC is reliable and robust in its ability to search for optimal solutions, making it a valuable algorithm for procedural content generation applications.

Although PSO#2 achieved a slightly higher best fitness value of 115.86 compared with PSO#1 and ABC, its performance remains competitive. However, the moderate standard deviation of 23.17 suggests some variability in fitness values across iterations, indicating slightly less stable performance compared to PSO#1 and ABC. Nonetheless, PSO#2's ability to effectively explore the solution space and converge towards near-optimal solutions makes it a viable choice for procedural content generation tasks.

However, the GA with the ``Mix'' selection approach achieved the best fitness value of 121.75, of GA variations indicating its ability to find relatively good solutions for the given task. However, the relatively high standard deviation of 52.35 suggests considerable variability in fitness values across iterations, indicating less stable performance compared to PSO#1, ABC, and PSO#2. Despite this variability, GA (Mix) still demonstrates the potential for generating diverse and engaging content in procedural content generation applications.

Move to GA (Tournament), This variant of the Genetic Algorithm achieved a best fitness value of 137.74, which is higher than those of all other selected algorithms in this study. Although the standard deviation of 25.28 suggests moderate variability in fitness values, a higher best fitness value indicates inferior performance compared with PSO#1, ABC, and PSO#2. The GA (Tournament) may still offer value in certain contexts but may require further optimization for optimal performance in procedural content generation tasks.

The last version of GA is GA (Biased) which showed the highest best fitness

value of 304.30 among all algorithms in this research, the GA with the ``Biased'' selection approach demonstrates the poorest performance in finding optimal solutions. The moderate standard deviation of 46.13 suggests some variability in fitness values, indicating a relatively stable but suboptimal performance compared with other algorithms. While the GA (biased) may have limitations in its current form, further experimentation and optimization efforts could potentially improve its performance in procedural content generation tasks.

Analysis of the results revealed valuable insights into the performance of various metaheuristic algorithms in procedural content generation. PSO#1 and ABC emerged as top performers, demonstrating superior performance in finding optimal solutions with high stability and consistency. Meanwhile, GAs show potential, but may require further refinement for optimal performance. Understanding the strengths and weaknesses of each algorithm is crucial for selecting the most suitable approach for specific procedural content generation tasks.

(PSO) excels in finding optimal solutions by simulating the behavior of a swarm of particles moving through a solution space. This algorithm is particularly effective for procedural content generation (PCG) tasks because of its ability to efficiently explore the solution space and converge towards optimal solutions.

This finding opposes established research [171] and lacks sufficient justification for the prevalent adoption of Genetic Algorithms (GAs) in procedural content generation (PCG) across various domains, particularly in game-level generation. Search-based PCG, a specific subset within PCG, is commonly addressed using generate-and-test methodologies such as genetic algorithms [172]. These studies collectively validate the utility and efficacy of genetic algorithms for PCG, emphasizing their capability to produce diverse and complex contents.

8.3.2 The Efficacy of Metaheuristics in Procedural Map Layout Generation

In addressing Research Question RQ4: ``In the context of map generation for PCG tasks, how are Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compared in terms of solution quality and convergence speed?" Chapter 7 [170] presents a thorough analysis to evaluate the effectiveness of different metaheuristic algorithms in procedural content generation (PCG), specifically focusing on map creation.

This comprehensive investigation encompasses a detailed examination of various algorithms, including Genetic Algorithms (GA) with diverse selection approaches (mix, biased, tournament), Particle Swarm Optimization (PSO) with two different values of cognitive parameters (C1), and Artificial Bee Colony (ABC). In this section, we provide insights into the performance and suitability of these algorithms for PCG map generation through precise experimentation and evaluation.

Although Chapter 5 [151] demonstrated that PSO performs well in race track generation, its performance in map generation exhibits a distinct pattern in the resulting map layouts, as shown in Figure 8.3. In contrast, the GA and its variants achieved better performance, producing high-quality maps. This observation indicates how the effectiveness of different algorithms can vary significantly across diverse PCG tasks, highlighting the dynamic nature of algorithm performance. It is important to remember that higher fitness values generally correspond to higher quality maps.

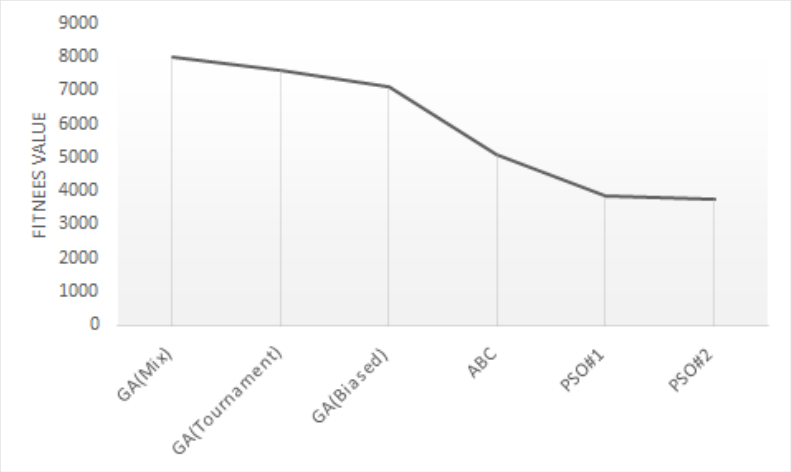


Figure 8.3: The performance of different algorithms in generating the map layout

Table 8.2 serves as a fundamental resource for recognizing the efficacy and stability of the selected algorithms deployed within the domain of the PCG in generating map layouts. Within this context, the table captures the essential performance metrics, notably the best-case fitness value and standard deviation (SD), offering critical insights into the capacity of the algorithms to generate optimal solutions for the PCG task.

	Best	SD
GA(Mix)	7998	807.312
GA(Tournament)	7626	535.23
GA(Biased)	7110	631.17
ABC	5110	315.10
PSO#1	3835	1083.09
PSO#2	3780	997.53

Table 8.2: Performance of the Algorithms in Map Task generation

An important finding from Table 8.2 is that the GA(Mix) consistently achieved the highest fitness value in the best-case scenario, demonstrating its efficiency in generating solutions close to the optimal outcome. With an impressive best fitness value of 7998, GA(Mix) has emerged as a compelling candidate for delivering high-quality PCG solutions. Furthermore, the algorithm's comparatively high standard

deviation of 807.312 suggests that the GA(Mix)'s performance might be less consistent across repetitions compared to the other algorithms. Further investigation is needed to determine the impact of this inconsistency on the quality of the generated maps and their suitability for PCG tasks.

The GA(tournament) and GA(biased) exhibited excellent performance characteristics. They achieved high fitness values of 7626 and 7110, respectively. Additionally, their standard deviations of 535.23 and 631.17 indicate relatively stable performance, suggesting consistency in generating results. This stable performance with good fitness values suggests a potential balance between exploration (finding new possibilities) and exploitation (focusing on promising areas) within the tournament (GA) and biased (GA) optimization processes. While neither achieved the highest absolute fitness value, these algorithms might be well-suited for scenarios where consistent and reliable map generation is crucial.

In contrast, PSO#1 and PSO#2 exhibited lower performances than the other algorithms. Their best fitness values were 3835 and 3780, respectively, and their standard deviations were considerably higher (1083.09 and 997.53). These results indicate that both PSO variants displayed inconsistencies across ten repetitions. Although PSO has demonstrated effectiveness in other PCG tasks, such as generating race tracks in Chapter 5, its performance in map generation appears to be less efficient and reliable compared to the GA variants explored in this study. Further investigation is required to understand why PSO struggles in this specific context.

Our findings regarding the performance of PSO in map generation appear to contradict the previous research on Evolutionary Game-based PSO (EGPSO) by Liu et al. (2008) [173]. EGPSO effectively addresses premature convergence and achieves robust convergence by incorporating an evolutionary algorithm into the PSO framework. However, in this study, we utilized a standalone PSO variant for a direct comparison with the GA and ABC algorithms.

While ABC's performance resulted in maps of average quality (fitness value of 5110), it exhibited good stability in terms of the standard deviation. The standard deviation, at its lowest value of 315.10, indicates consistent results when generating

map layouts. This suggests that ABC may be a reliable choice for applications in which consistent map generation is important. Furthermore, as highlighted in Chapter 3 of our previous study [50], the application of the ABC algorithm to PCG tasks remains unexplored. This underscores the novelty of our approach in utilizing ABC for map generation, and contributes valuable insights to the field.

The performance and features of GA, PSO, and ABC used in map layout generation are thoroughly examined in Chapter 7 [170]. This analysis helps researchers and game developers to choose the most suitable algorithm for their specific needs and goals in different PCG tasks. These results serve as a foundation for further research. Combining these algorithms, each with their unique strengths, holds promise for the future. By carefully modifying variables, such as crossover frequency, mutation rate, and population size, we can create more effective and adaptable content creation techniques.

8.4 Comparative Analysis of Algorithms Performance

While Chapter 5 [151] focuses on evaluating the algorithm performance specifically in race track generation, Chapter 7 [170] expands our understanding of map generation. These chapters provide valuable insights into the application of the algorithms studied in our research to different tasks. This section aimed to answer the following two research questions:

- **RQ5: How stable are the solution quality and convergence speed of Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) when applied to race track and map applications?**
  This basic question seeks to determine the extent to which the results of the comparative analysis are relevant. Our specific goal is to determine the extent to which the knowledge we have learned from our investigation of the creation of race tracks and maps can be applied to other procedural content generation (PCG) tasks. Solving this important question creates innovation opportunities. Imagine using the optimised algorithms found in our preliminary research to

craft missions specific to each player's interests, add unique species to immersive environments, or develop dynamic stories. By extending our investigation beyond our original emphasis, we open the possibility of offering developers a flexible toolkit that can handle various facets of procedural content production.

- **RQ6: How applicable are the findings of this comparative study to a broader range of PCG tasks beyond those specifically examined, such as race track and map generation ?**

  This research question has shifted our attention to its broad applicability to adaptability. It seeks to understand how well metaheuristic algorithms perform when faced with the challenges of diverse procedural content generation (PCG) tasks. Will algorithms that excel in race track generation demonstrate the same effectiveness in creating other PCG tasks? Examining their performance across various applications is crucial. Exploring these questions lays the foundation for the development of robust and adaptable algorithms that can thrive in dynamic and evolving environments.

Our research indicates an intriguing pattern in the diversity of performance, depending on particular situations. Although PSO#1 and ABC performed well in both race track and map generation tasks, the order of their performance changed. This highlights the dynamic nature of the algorithm performance across different PCG domains. ABC performed better than PSO in map generation.

In contrast, PSO offers a more complex view. Although it performs competitively on the race track, its ranking is the worst and its map generation standard deviation suggests possible generalizability issues. This emphasises how the algorithm performance is dynamic and why moving them between other PCG applications requires cautious consideration.

The GA variations provide even more uncertainty. GA(tournament) performs consistently, suggesting that it may be able to handle a variety of tasks. However, GA(biased) shows a trade-off: it generates maps with less variability but at the expense of overall fitness. This emphasises the necessity for further research on their

potential for optimization and flexibility in a range of situations.

In the following subsections. We compared the performances of these algorithms on two PCG task race tracks and map generation in detail. We also evaluated their ability to create these tasks and how well they adjusted various evaluation metrics. This in-depth examination highlights the strengths and weaknesses of each algorithm, guiding future research and advancing the PCG strategies.

8.4.1 Genetic Algorithm (GA) Performance

While Genetic Algorithms (GA) exhibited lower performance in race track generation (as discussed in Chapter 5), they demonstrated good results in map layout generation (Chapter 7). This highlights the significant differences in the efficiency and effectiveness of GA depending on the specific PCG task.

Table 8.3 compares the performance of the three variations of GA applied to the task of generating race tracks. Each variation, GA(mix), GA(tournament), and GA(biased), was evaluated based on three criteria: worst fitness value, best fitness value, and standard deviation.

The worst fitness value represents the highest fitness score achieved by the GA over ten iterations. In the context of this task, higher values are generally undesirable, because they indicate that the algorithm generated tracks further away from the desired outcome. However, the best fitness value represents the lowest fitness score achieved, which is desirable because it signifies the generation of a track that is closer to the desired outcome. Finally, the standard deviation measures the variation in the fitness values across ten iterations. A lower standard deviation generally indicates more consistent performance.

	GA(Mix)	GA(Tournament)	GA(Biased)
Worst Fitness Value	299.47	225.63	443.28
Best Fitness Value	121.75	137.74	304.30
Standard Deviation	52.35	25.28	46.13

Table 8.3: Comparison of GA variation in the Task of race track

Based on the results from Chapter 5, we can see that GA(Mix) achieved the best fitness value (121.75) among the three variations. However, it also exhibited a relatively high worst fitness value (299.47) and high standard deviation (52.35). This suggests that, although it occasionally generates very good tracks, its overall performance is inconsistent.

The GA(Tournament) shows the lowest fitness value (225.63) in the worst fitness scenario compared to both GA(Mix) and GA(Biased). It also achieved the second best fitness value (137.74). In addition, it exhibits the lowest standard deviation (25.28) among the three, indicating the most consistent performance across ten iterations.

Among the three variations, the GA(biased) showed the worst performance in terms of both the worst fitness (443.28) and best fitness (304.30). This indicates a tendency to consistently generate tracks that deviate significantly from the optimal solutions. Although its standard deviation (46.13) was moderate, it was not sufficient to compensate for the overall lower quality of the tracks.

While GA(Tournament) emerges as the most promising choice based on minimizing both the worst and best fitness scores in ten iterations, further investigation is necessary. The limited number of iterations and specific problem definition require additional testing with more iterations and a deeper understanding of the optimization goals to draw more robust conclusions.

However, it is clear that the choice of GA variant for race track generation depends on the specific priorities of the task if we are aiming for the absolute best possible outcomes, even if inconsistent GA(Mix) might be suitable, despite its occa-

sional poor runs. If consistent performance and reliable track quality are essential, then the GA(tournament) could be a better choice, even if it might not always reach the absolute best tracks. To achieve a balance between potential and consistency, GA(Biased) offers a middle ground, but further analysis is required to determine its suitability.

For map generation, Table 8.4 presents the performance of the three GA variants applied to this task. Higher fitness values indicate better results in this context. Each variant exhibits unique strengths and weaknesses, offering valuable insights into the trade-off between exploration and exploitation in the context of this optimization process.

	GA(Mix)	GA(Tournament)	GA(Biased)
Worst Fitness Value	5236	5720	5040
Best Fitness Value	7998	7626	7110
Standard Deviation	807.312	535.23	631.17

Table 8.4: Comparison of different algorithms in the task of Map layout

GA(Mix) emerges as the leader in terms of absolute best fitness score, reaching an impressive mark of 7998. However, this achievement comes at cost. The high standard deviation of 807.312 suggests occasional outliers and potentially less consistent performances across generations. This indicates that while GA(Mix) can occasionally generate exceptional maps, its overall performance might be less predictable.

In contrast, the GA(Biased) prioritizes consistency, as evidenced by its significantly lower standard deviation of 631.17 This translates to a more consistent performance across generations, with fewer instances of significant variation in map quality. However, the trade-off lies in the lower best fitness score of 7110. This suggests that GA(biased), while offering consistent performance, might sacrifice the potential to find remarkable map combinations.

Finally, the GA(Tournament) effectively balanced the two extremes. Its best fit-

ness score of 7626 remains competitive, demonstrating its ability to generate high-quality maps. Additionally, its standard deviation of 535.23 sits comfortably between those of the other two variants, indicating a respectable level of consistency without sacrificing the potential for exceptional outcomes. This suggests that GA(Tournament) offers a promising compromise between exploration and exploitation, making it a strong candidate for generating diverse and consistently high-quality maps.

This comparative analysis highlights the importance of considering both the best fitness score and the standard deviation when evaluating the performance of the optimization algorithms. While striving for the highest possible fitness scores is desirable, achieving consistency and avoiding oddities are also crucial for ensuring reliable and predictable outcomes in the context of map generation.

A comparison of Tables 8.3 and 8.4 reveals potentially significant similarities in the GA performance between tasks. This emphasizes the importance of considering problem-specific characteristics when selecting the algorithms. Understanding the evaluation criteria is crucial for interpreting results.

The tables highlight how different GA selection methods (Mix, Biased, Tournament) excel in distinct ways. This emphasizes the crucial role of considering specific characteristics of the PCG problem when selecting a metaheuristic. Different algorithms might have strengths and weaknesses suited to various goals (e.g., exploration vs. exploitation and consistency vs. occasional high peaks).

In terms of exploitation and exploration, the performance differences between GA variants reveal a delicate balance between exploration and exploitation in metaheuristics. GA(Mix) achieves high peak scores, but suffers from inconsistency, suggesting that it can explore diverse possibilities more readily. GA(Biased) prioritizes exploitation, leading to more consistent results but potentially missing exceptional solutions. GA(Tournament) strikes a balance, offering a compromise between exploration and exploitation. This trade-off is relevant across various metaheuristics, impacting the solution quality and diversity. Table 8.5 lists the key findings for the

two PCG tasks.

	Race Tracks	Map Layouts
GA(Tournament)	Emerging as the most suitable option, achieving a balance between the best and worst fitness values while displaying the most consistent performance (lowest standard deviation)	Again proved to be a strong contender, balancing the best fitness with a reasonable standard deviation.
GA(Mix)	This demonstrated the potential to generate exceptional tracks but lacked consistency.	Showcased potential for highly rated maps but with potential inconsistencies.
GA(Biased)	Prioritized consistency but compromised the potential for achieving exceptionally good tracks	Exhibited consistent performance but achieved a lower best fitness compared to other variants.

Table 8.5: Key Findings in GA variations for the both applications

Selecting the most suitable metaheuristic algorithm requires a comprehensive understanding of the objectives of the problem, the available resources, and the inherent constraints. Evaluating performance using relevant metrics aligned with the project's goals is crucial.

Further studies have the potential to illuminate the reasons for the observed variations in the metaheuristic performance. More flexible and effective algorithms may be created by examining the principles underlying different Genetic Algorithm (GA) selection strategies and how they interact with one another in certain Procedural Content Generation (PCG) tasks. This is consistent with the focus of our research on metaheuristic comparisons, which highlights the need for comprehensive assessments beyond performance measurements.

Several factors may have contributed to the observed performance variations across the tasks. For example, GA(Tournament) [174], [175] prioritizes selection

based on individual fitness values, potentially leading to effective exploitation of promising areas within the search space. Although this approach facilitates the swift identification of favorable solutions, it may encounter limitations in exploring diverse regions and escaping local optima.

By contrast, GA(Biased) [176], [177] potentially incorporates a predefined bias in its selection process, favoring solutions exhibiting specific characteristics. While advantageous if aligned with the desired outcomes, it can impede exploration and lead to suboptimal solutions when misapplied.

GA(Mix) attempts to strike a balance between exploration and exploitation by combining tournament selection with a biased source. Tournament selection encourages the identification and propagation of high-fitness individuals, whereas bias can guide a search for promising regions. However, the effectiveness of this balance depends on the specific characteristics of the biased source and problem context.

Furthermore, the effectiveness of GA(mix) is highly dependent on the quality and relevance of the incorporated bias. A bias aligned with desired solutions can be advantageous, whereas irrelevant or misleading biases may result in inferior performance compared to GA(tournament).

Regarding parameter settings, even with identical selection mechanisms such as tournament and biased selection, the specific parameter configurations for each GA variant can influence their behavior. Factors such as tournament size or selection pressure in GA(tournament) and the nature and impact of the bias in GA(mix) can influence the search process and contribute to performance disparities.

8.4.2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) stands one of the evaluated algorithms owing to its unique performance attributes, prompting the need for further exploration. This section aims to delve into the behavior of PSO across their applications race track and map generation, drawing upon the insights gained from the available data

However, particle swarm optimization (PSO) is a powerful evolutionary algorithm inspired by the collective movement of birds or fish, where individual particles update their positions based on their own experience and the best experiences of their neighbors [178]. Two key parameters, C1 and C2, significantly influence the behaviour and performance of the PSO [179] and [180].

- **C1 (Cognitive Parameter): This parameter controls the effect of a particle's personal best position on its velocity update. A higher C1 value emphasizes the particle's own experience, potentially leading to faster exploration of the search space, but also increasing the risk of getting stuck in local optima. Conversely, a lower C1 value prioritizes learning from the global best position, promoting convergence towards the optimal solution, but potentially hindering exploration.**

- **C2 (Social Parameter): This parameter governs the influence of the neighborhood best position on the particle's velocity update. Similar to C1, a higher C2 value encourages exploration by emphasizing the best solution found within a local neighborhood. Nevertheless, a low C2 value prioritizes the global best, potentially leading to premature convergence and missing better solutions in unexplored regions.**

This study highlights the swift convergence capabilities of both PSO implementations, particularly when the cognitive parameter (C2) was set to four in PSO#1 and 5 in PSO#2. Across both race track generation and map layout tasks, these PSO variants consistently achieved the target fitness value within only eight iterations. This significantly outperformed the convergence speeds observed in the Genetic Algorithm (GA) and Artificial Bee Colony (ABC) algorithms employed in this study. Such rapid generation of initial solutions makes PSO a highly attractive option for PCG applications, where fast prototyping or iterative design methodologies are crucial.

The optimal values of C1 and C2 can vary depending on the specific problem

and desired balance between exploration and exploitation. In our study, setting C2 to 4 in PSO#1 and 5 in PSO#2 likely facilitated rapid convergence by promoting exploration within a local neighborhood while still being influenced by the global best solution. This combination may have allowed PSO to quickly identify promising regions and converge to optimal solutions within only eight iterations.

While boasting impressive initial convergence, the PSO final fitness results varied across the two PCG tasks. In the race track scenario, PSO#1 achieves the best average fitness, whereas PSO#2 exhibits a slightly poor performance, but lacks a significant difference from PSO#1, see Table 8.1. In the map layout application, both PSO#1 and PSO#2 final fitness values fell short of the best GA and ABC results, see Table 8.2. Although not always reaching the top spot, PSO remained competitive in terms of the final fitness in both applications, suggesting its potential for generating high-quality solutions.

Unlike PSO, Genetic Algorithms (GA) rely on crossover and mutation operators to explore the search space, which can be slower than PSO's direct velocity updates. Similarly, Artificial Bee Colony (ABC) uses a combination of exploration and exploitation phases, which might not converge as quickly as PSO in certain scenarios.

One caveat associated with PSO performance was its higher variability across multiple runs compared to the GA (tournament) in both applications. This indicates that achieving results consistent with PSO may require careful parameter tuning or additional runs. This variability poses a challenge, as designers may seek algorithms with more predictable outcomes, particularly for critical PCG tasks.

Despite the differences in the problem domains (track race versus map layout), PSO displays remarkably similar convergence patterns and performance characteristics. This suggests that its behavior may be consistent across various PCG tasks, potentially making it a versatile tool for different content generation requirements (see Table 8.6.

	PSO#1 Rank	PSO#2 Rank
Race track Application	1	3
Map layout Application	5	6

Table 8.6: PSO Best fitness rank among the investigating algorithms across two application

This study hints at the influence of the C2 coefficient on PSO performance. Exploring different C2 values or adjusting other parameters could potentially enhance PSO's final fitness and address the observed variability in both applications. Further research efforts focused on parameter tuning could unlock even greater potential from PSO in the realm of PCG.

In addition, PSO performance is characterized by its stability and consistency across iterations. The low standard deviation observed in its fitness values indicates minimal variability in performance, reflecting PSO's reliability and robustness of PSO in consistently determining near-optimal solutions consistently, see Table 8.7.

	PSO#1 Rank	PSO#2 Rank
Race track Application	5	3
Map layout Application	6	5

Table 8.7: PSO SD rank among the investigating algorithms across two application

Overall, PSO's ability to efficiently explore the solution space, adapt to varying task requirements, and maintain stable performance makes it a highly effective algorithm for procedural content generation tasks. Its superior performance in terms of both optimization quality and stability positions PSO as the top performer among the algorithms evaluated in this study.

Our study emphasizes the importance of customizing C2 values for PSO in PCG tasks, where the optimal values depend on the complexity of the problem and the desired quality of the solution. Adjusting C1 and C2 dynamically throughout the optimization has the potential to further enhance performance. Additionally, integrat-

ing PSO with other algorithms can capitalize on their strengths, potentially leading to better outcomes. Understanding the roles of C1 and C2 enables effective tuning of PSO across various PCG tasks, promoting rapid convergence and high-quality content generation.

8.4.3 Artificial Bee Colony (ABC)

An Artificial Bee Colony (ABC) is a nature-inspired metaheuristic algorithm that mimics the foraging behavior of honeybees to solve optimization problems. This involves three types of bees [181]:

- **Employed Bees: Explore the neighborhood of known food sources (solutions) and exploit their quality.**
- **Onlooker Bees: Evaluate the quality of food sources presented by employed bees and potentially join the exploitation process.**
- **Scout Bees: Randomly explore the search space to discover new food sources and replace abandoned ones.**

However, Chapters 5 and 7 [151] and [170] offer valuable insights into the performance of the Artificial Bee Colony (ABC) algorithm for procedural content generation (PCG), specifically in track races and map layouts.

Unlike the PSO rapid initial burst, ABC exhibited a slower initial convergence in both applications. It reaches the threshold fitness value later than the other algorithms, indicating that its exploration phase requires more iterations. However, this was followed by continuous improvement throughout the remaining iterations, leading to impressive final fitness results. This ``slow and steady'' approach may be suitable for scenarios in which thorough exploration and gradual refinement are desired.

In both track races and map layouts, the ABC scheme achieved the highest fitness score among all the evaluated algorithms. It consistently found exceptional solutions, indicating its potential to generate highly optimized and desirable track

layouts and maps. This makes it a strong contender for applications in which finding the absolute best outcome is paramount (see Table 8.8.

	ABC Rank
Race track Application	2
Map layout Application	4

Table 8.8: Ranking of Best Fitness Achieved by ABC Compared to Other Investigated Algorithms Across Two Applications

While achieving the best fitness, ABC also displayed the worst fitness score, particularly in the race track scenario. This suggests that the ABC occasionally converges to suboptimal solutions. Additionally, its standard deviation, indicating variability across runs, was moderate in both applications, falling between those of the GA and PSO in terms of consistency. This trade-off between potentially achieving the best possible solution and the risk of inconsistency, along with slower initial convergence, necessitates careful consideration when choosing ABC for PCG tasks.

Despite the moderate variability, the results emphasize ABC's consistent performance across multiple runs. This stability can be valuable for designers seeking reliable results, particularly when generating critical game content.

Unlike GAs and PSO, ABC employs a unique population-based search inspired by the foraging behavior of honey bees. This approach offers a different perspective on problem solving, potentially making it effective in scenarios where other algorithms struggle.

Further research delving deeper into the inner workings of the ABC algorithm, particularly the influence of the parameters and colony dynamics, could reveal strategies for improving its performance and consistency. Exploring how parameters, such as the number of employed bees or onlooker bees, affect their search behavior could be a fruitful avenue for future investigation.

ABC presents a unique value proposition for PCG tasks. Its ability to consistently find exceptional solutions, especially the highest fitness scores, makes it a potential game changer. However, the trade-off for this excellence lies in its slow initial convergence, moderate variability, and possibility of converging to suboptimal solutions. Understanding its colony dynamics and exploring parameter tuning could unlock further potential and address these limitations, making ABC an even more attractive option for diverse PCG applications (Table 8.9.

	ABC Rank
Race track Application	1
Map layout Application	1

Table 8.9: Ranking of CD Achieved by ABC Compared to Other Investigated Algorithms Across Two Applications

ABC is a compelling candidate for PCG tasks owing to its tripartite balance between exploration and exploitation, adaptive capabilities, and inherent robustness. Despite limited exploration of its application in PCG by previous scholars [50], our study endeavors to thoroughly investigate its potential and limitations.

8.5 Task-Specific Considerations

According to Shaker(2016) [1] and Sturtevant (2018) [3], the procedural content generation (PCG) field is defined by a wide range of approaches and strategies. The main focus of these studies was to highlight the diversity of available methodologies; this is a subject that is also present in our research, highlighting the lack of a single algorithm that is suitable for all PCG problems. This alignment is consistent with the observed variations in algorithms such as Genetic Algorithms (GAs) performance. Although GAs are effective in many situations, they have drawbacks, particularly when it comes to tasks such as race tracks. Hence, other approaches like (ABC) and (PSO) should be investigated. Accepting this diversity of methods could lead to the creation of more complex and efficient solutions adapted to the specific needs of each PCG situation.

The features of a given task have a significant effect on the effectiveness of the algorithm. For instance, PSO excels at the exploration stage of building a racecourse, whereas ABC focuses on precise track design. GA is far more adept at navigating complicated map layouts than PSO when it comes to map creation. These findings demonstrate the importance of understanding the particulars of quality standards, complexity of cases, and other distinctive features.

When selecting an algorithm, tradeoffs are always involved. PSO's rapid exploration may yield results more quickly; however, the quality of the solutions may suffer. However, development may progress more slowly as a result of ABC's meticulous methodology of ABC. A thorough understanding of these trade-offs facilitates an informed decision-making process customized to the specific priorities of the PCG task.

As such, a ``Task-Specific Considerations'' approach promotes adopting a context-aware selection framework in place of a ``one-size-fits-all'' approach. The best algorithmic decisions may be made to produce the desired results by carefully examining the specifics of each PCG, identifying the advantages and disadvantages of the available algorithms, and assessing the related trade-offs. Beyond the discussed algorithms, further research efforts could reveal new metaheuristic or hybrid approaches, thereby expanding the range of possibilities for various PCG tasks.

Moreover, the algorithmic performance is significantly affected by the manner in which the problem is described and the definition of the evaluation metrics. It is crucial to ensure that these elements represent the intended goals of the PCG task precisely.

Finally, utilizing domain knowledge relevant to a particular PCG domain, such as the genre of games or content type, can provide invaluable insights into the relevant task attributes and possible algorithmic fit.

8.6 Summary

This study explored various algorithms for PCG, highlighting that selecting the best approach requires careful consideration of task-specific factors. The optimal choice depends on the unique characteristics of the PCG task, desired outcomes, and potential trade offs involved. Our findings offer valuable insights into the performance of context-dependent algorithms when applied to different optimization tasks. While PSO demonstrated exceptional robustness when applied to generate race tracks, its initial exploration phase frequently produced high-quality tracks, suggesting potential room for further optimization. This aligns with the observed performance of the GA(tournament) in the previous section, where it effectively balances exploration and exploitation. However, when time constraints were not a critical factor, the thorough and consistent exploration of Artificial Bee Colony (ABC) proved advantageous in comprehensively exploring the search space for potential track configurations.

When the task was shifted to generating maps, PSO displayed poor performance in handling the complexities of map layouts. However, Artificial Bee Colony (ABC) has emerged as a moderate choice, achieving faster convergence and generating high-quality map content. These findings highlight the potential of ABC as an effective tool for developing complex maps.

These opposing findings raise the question of the assumption that GA dominates PCG universally and motivate us to adopt a context-aware approach to algorithm selection rather than a single one. Each algorithm has its advantages and disadvantages. When faced with time constraints, PSO's quick first investigation proves to be helpful; however, ABC's cautious approach excels when faced with strict quality criteria. Realizing these trade-offs and appropriately adjusting the selection allows PCG to reach its maximum potential.

This chapter highlights the significance of a careful study of each PCG task considering the complexity, quality requirements, and desired solution speed. It recommends a ``Task-Specific Consideration'' approach. Moreover, it promotes re-

search beyond conventional Genetic Algorithms (GAs) and highlights the distinct advantages of Artificial Bee Colony (ABC) and Particle Swarm Optimisation (PSO). Concerning exploration, exploitation, consistency, and solution quality, each algorithm has unique benefits and drawbacks, and wise decisions should be made in accordance with the project-specific requirements.

Context is essential. Every PCG work requires a unique analysis that uses domain expertise, such as game genre and content type, to improve algorithmic fit. Most importantly, particular algorithmic strengths and weaknesses depend on the properties of the tasks themselves; therefore, future research should be guided by study limitations within this chapter, such as the specific tasks investigated.

Through the integration of these perspectives, researchers and developers will be able to make knowledgeable choices about the algorithm that optimally addresses their distinct PCG requirements, finally producing engaging material that is customized to their particular goals.

Chapter 9

Prospects, Limitations and Conclusions

This thesis embarked on a journey to unveil the efficacy of various metaheuristic algorithms in the realm of Procedural Content Generation (PCG) for video games by delving into a comparative analysis of Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC). Through exhaustive experiments comparing Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC), this study gauged the effectiveness and efficiency of these algorithms in producing game maps. The evaluation metrics encompassed the convergence speed and content quality.

Our research comes close to this last chapter, which summarizes the key findings, considers their broader relevance, acknowledges any inherent limitations, and suggests future research directions.

9.1 Summary of Findings

The primary goal of this study was to address the following research topics.

- **RQ1: What are the most common and effective meta-heuristic algorithms used for procedural content generation (PCG) in games and how have these algorithms been applied across various domains within PCG?**
- **RQ2: What are the main challenges and limitations of using meta-heuristic algorithms for PCG and how can these be addressed in future research?**
- **RQ3: How do Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compare in terms of solution quality and convergence speed when applied to the task of race track generation in PCG?**
- **RQ4: In the context of map generation for PCG tasks, how are Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) compared in terms of solution quality and convergence speed?**
- **RQ5: How stable are the solution quality and convergence speed of Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) when applied to race track and map applications?**

- **RQ6: How applicable are the findings of this comparative study to a broader range of PCG tasks beyond those specifically examined, such as race track and map generation?**

This study aimed to evaluate various metaheuristic search algorithms for their performance in procedural content generation (PCG) tasks. Despite the widespread use of Genetic Algorithms (GAs) in PCG, there is increasing acknowledgment in the literature of the potential advantages offered by alternative algorithms. Our evaluation primarily employed a quantitative approach, comparing the efficacy and efficiency of these algorithms based on their ability to discover high-quality solutions using fitness functions and the speed of convergence.

Because the selected algorithms are stochastic, our evaluation involved generating multiple game levels and employing statistical methods to identify the significant differences. By keeping all variables constant, except for the algorithm, we could accurately attribute any observed differences to algorithmic variations. It is important to note that playability was intentionally omitted from our quantitative evaluation because it is already incorporated into the fitness function used to evaluate each particular game level.

The comparative approach we adopted utilized GA, ABC, and PSO as metaheuristic algorithms across two distinct PCG tasks: race track generation and map generation. By using quantitative analysis for objective performance metrics and comparing the outcomes of both tasks, we aimed to gain a comprehensive understanding of the quality and suitability of the generated content for gaming.

Our research design followed a mixed-methods approach, sequentially addressing the research questions through a systematic literature review, algorithm implementation, and evaluation. The systematic literature review laid the foundation for the subsequent stages, whereas the implementation and evaluation stages provided insights into the effectiveness of the selected algorithms in generating game content.

By dynamically adjusting the algorithm parameters and combining different

metaheuristic approaches, we sought to further enhance algorithm performance. Our findings contribute to the field by providing insights into the strengths and limitations of various metaheuristic algorithms for PCG tasks, paving the way for future research in this domain.

To answer RQ1 and RQ2, we used a systematic method, following the guidelines set by Kitchenham and Charters (2007) [73], to thoroughly explore the existing literature in the field. Relevant research papers were identified and analyzed through careful planning, searching, and reporting. The main focus was on assessing how meta-heuristic approaches are applied in procedural content generation (PCG), and tackling the challenges involved in using these algorithms. A wide range of metaheuristic algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), have been examined in 97 studies. The goals were to identify gaps, assess fitness and content quality, explore challenges, and suggest future areas of study.

The analysis of 97 selected studies revealed a strong focus on evolutionary approaches, particularly GA in PCG. However, this bias towards the GA does not negate the potential of other metaheuristic algorithms to generate game content. Although GA demonstrate diverse applications in PCG across different genres, SA has seen limited usage, and PSO has been discussed and utilized in specific research papers. Furthermore, existing literature provides limited evidence of alternative metaheuristic approaches, suggesting the need for further exploration to enhance the diversity and effectiveness of PCG methods beyond GA.

Initially, the study examined how various metaheuristic algorithms were used in the PCG (RQ1) and their prevalence. Although Genetic Algorithms (GA) have been widely used, particularly for evolutionary PCG approaches, our study highlights the potential of other metaheuristic algorithms in content generation tasks. Although the GA showed versatility across different PCG contents, the utilization of SA was limited to one study, whereas PSO was discussed in six papers, focusing on content creation, animation modules, and game balance enhancement. The limited evidence

from alternative approaches underscores the need for further exploration to diversify and improve PCG methods beyond GA.

Our study concludes by discussing opportunities, challenges, and future research directions for PCG using metaheuristics (RQ2). They stressed the importance of exploring alternative metaheuristic approaches, enhancing the fitness function quality, integrating algorithms, promoting diversity, considering user experience, and establishing robust evaluation metrics. Ethical considerations were also identified as crucial and required further investigation. The analysis revealed gaps in addressing these aspects, highlighting the need for future research to advance PCG and fill existing knowledge gaps. Suggestions for future research include developing new algorithms, exploring hybrid approaches, and examining the ethical and social implications of PCG.

By answering these research questions (RQ1) and (RQ2), we offer a comprehensive overview of how metaheuristic algorithms are used in procedural content generation, especially in computer games, in Chapter 3. It serves as a valuable resource for researchers and game developers seeking to use optimization techniques to create high-quality content in video games and other digital media.

Investigating the answers to RQ1 and RQ2 led us to develop RQ3 and RQ4 and find answers for them. However, to answer RQ3, we evaluated the performance of various metaheuristic approaches in PCG tasks, specifically focusing on race track generation, to challenge the predominant use of GAs in Chapter 5. These findings highlight the performances of the three algorithms. Notably, the study revealed that the selection strategy significantly affected the GA performance and repeatability.

Comparatively, the PSO algorithm exhibited faster convergence than the GA, with proper tuning potentially yielding comparable or slightly improved repeatability. A common approach involves adapting the inertia weight (ω) during optimization to balance exploration and exploitation, thereby emphasizing the need for experimentation and tuning to determine the optimal value of a given problem. Conversely, the

ABC algorithm demonstrated high repeatability at the expense of slower initial convergence, yet both PSO and ABC showed promise in finding better-quality solutions overall.

Moreover, the results suggest that researchers and practitioners in search-based procedural content generation (SBPCG) may benefit from considering alternative algorithms rather than solely relying on GAs because of their popularity. However, it is crucial to recognize that no single algorithm reigns supreme, and the selection process should involve trade-offs based on the algorithm's characteristics and the specific problem at hand. For instance, although PSO exhibited rapid initial convergence, its solution quality improvement was not guaranteed, making it suitable for scenarios with low computational costs and time constraints. Conversely, ABC may be preferable when time is less of a constraint, aiming for high-quality solutions with confidence, despite slower convergence.

Although existing comparative studies have provided insights into algorithm performance across various domains, the current focus remains limited to procedural content generation for video games. This study underscores the need for further comparative research to explore the performance characteristics of alternative algorithms in this context and beyond.

Answering this QR3 contributes to expanding our understanding of how different metaheuristic algorithms fare in PCG tasks, offering insights that could inform future studies and aid in the selection of appropriate methods to ensure that game designs meet both designer and player expectations.

To enhance our understanding of metaheuristics in PCG, we addressed RQ4 by evaluating the effectiveness of various optimization algorithms in map layout generation in Chapter 7. Our assessment included the GA (mix), GA (biased), GA (tournament), ABC, PSO#1, and PSO#2.

The results indicate that traditional GA implementations are superior to ABC

and PSO in terms of the map generation content quality. However, further research is required to explore how different algorithms, such as ABC, can be effectively integrated into the PCG workflow to achieve the best possible results for creating immersive and dynamic gaming experiences.

We found that the algorithm performance varies based on problem complexity and parameter settings, highlighting the need for further exploration and experimentation to identify the most suitable algorithm for specific tasks. These findings provide a foundation for future advancements in optimization algorithms.

Future endeavors could involve fine-tuning the algorithm parameters to enhance the performance. Experimentation with parameters such as population size, mutation rate, crossover probability, swarm size, and inertia weight in PSO may yield varied outcomes, thereby improving algorithm performance. Additionally, exploring alternative selection methods or operator variations within genetic algorithms can yield valuable insights into optimization processes.

Furthermore, conducting experiments on a broader range of benchmark problems can provide a more comprehensive understanding of algorithm performance across diverse problem domains. Testing algorithms on various problem sets allows the assessment of their robustness, scalability, and generalizability.

Addressing the limitations observed in this study, future research could focus on enhancing computational efficiency, exploring ways to reduce time complexity, and implementing parallelization techniques or hybridizing algorithms with other optimization methods. Additionally, comparative studies with state-of-the-art optimization algorithms or advanced metaheuristic techniques can offer deeper insights into their relative strengths and weaknesses.

to answer RQ5 and RQ6, we explored and discussed the outcomes in Chapters 3, 5, and 7. We explored the selected algorithms used in our study, emphasising the need to select the most suitable one for specific tasks. Our findings show that the

performance of these algorithms depends on the nature of the task, the desired outcomes, and potential trade-offs.

Our findings reveal distinct strengths and weaknesses of metaheuristics for PCG tasks. While GAs excel in map creation, they struggle to craft high-quality race tracks. Conversely, PSO demonstrates superior convergence speed and efficiently finds neer-optimal solutions for content-generation tasks. However, PSO may require further optimization for specific aspects, such as map content generation. This suggests that integrating PSO, with its efficient solution finding and potentially a well-optimized ABC for specific content generation, into the PCG workflow holds significant potential for creating engaging and diverse content experiences within video games.

Finally, it can be concluded that these findings challenge the notion that GA is universally the best choice for PCG tasks. Instead, it is crucial to consider each algorithm's strengths and weaknesses and choose accordingly based on specific project requirements.

Overall, this thesis underscores the importance of tailoring the algorithm selection for the task at hand. By understanding the unique characteristics of each algorithm and the tasks for which they are best suited, researchers and developers can make informed decisions to produce engaging content. This study initiates a discussion on the performance of optimization algorithms, laying the groundwork for future investigations. Subsequent research should concentrate on fine-tuning the algorithm parameters, exploring alternative operators and selection methods, testing algorithms on diverse problem sets, improving computational efficiency, and conducting comparative analyses with advanced optimization techniques. These endeavors will contribute to the ongoing development and enhancement of optimization algorithms to address complex real-world problems.

9.2 Implication and Future Direction

This thesis has significant implications for the advancement of PCG, an essential function in enhancing content production and developing immersive gaming experiences. Our study makes recommendations for future research by addressing the highlighted opportunities and challenges. These involve examining the ethical and social elements of PCG systems, exploring hybrid techniques that integrate various algorithms, and developing specialised metaheuristic algorithms for PCG tasks. Researchers and game developers can improve PCG techniques and raise the standards and applicability of generated content in video games and other digital media by exploring these fields. Implications and future directions are discussed in detail in the following subsections.

9.2.1 Enhancing Metaheuristic Algorithms

This study revealed a dynamic research environment that encourages additional investigation into the field of metaheuristic algorithms for Procedural Content Generation (PCG). Future research is necessary to fully comprehend the potential of the three algorithms studied: Genetic Algorithms (GA), particle swarm optimization (PSO), and Artificial Bee Colony (ABC). Each algorithm exhibits strengths and shortcomings during analysis.

Further research avenues present significant opportunities to enhance the performance of each algorithm. GA for instance, could benefit from exploring alternative selection methods such as elitism selection [182], alongside variations in crossover and mutation operators such as single-point vs. two-point crossover which could potentially enhance the effectiveness of optimization.

By contrast, PSO depends on an accurate balance between exploration and exploitation. Exploring the dynamic adjustments of the inertia weight, which controls particle-to-particle information transmission, presents promising opportunities. Furthermore, exploring other neighbourhood topologies, including ring or star arrange-

ments, could assist in guiding information flow more efficiently, resulting in quicker convergence and possibly improved solutions in intricate PCG environments.

Finally, ABC offers areas in which methods for choosing food sources should be improved. Evaluating the efficacy of various strategies, such as probabilistic selection or the roulette wheel, allows the most effective exploitation of practical solutions specific to PCG problems. Moreover, investigating alternative scout bee techniques could improve the ABC exploration performance, especially for applications requiring larger search regions. Considering these issues, ABC may be a useful tool for handling various complex PCG tasks.

By exploring these customised improvements for every algorithm, researchers can open the way to a new era of PCG that is marked by greater effectiveness, flexibility, and the production of engaging content. The possibility of procedurally generated environments and the future of game development are extremely promising; this exploratory expedition proves this.

9.2.2 Benchmark Problems

As discussed in Section 9.2.1, metaheuristic algorithms require refinement, and although these focused improvements show considerable potential, it is imperative to broaden the experimental scope by including additional benchmark problems. Testing algorithms across diverse problem sets is essential to assess their robustness, scalability, and generalizability, as algorithm behaviors often vary between different problem domains. This comprehensive evaluation will ensure the development of algorithms tailored not only to specific tasks but also adaptable to a wider range of PCG problems.

By exploring these tailored enhancements and conducting broader experiments, researchers can usher in a new era of PCG characterized by enhanced effectiveness, flexibility, and the creation of truly engaging content. The future of game devel-

opment and the boundless possibilities of procedurally generated landscapes hold great promise, as evidenced by this exploratory journey.

Comparative studies using cutting-edge algorithms offer invaluable insights, guide informed algorithm selection, and propel the entire PCG field to new heights [183]. By pursuing these avenues, we can unlock the full potential of diverse metaheuristics, empower game developers with sophisticated tools, and ultimately craft game worlds brimming by captivating content and ever-evolving experiences. Recall that innovation is key to unlocking the true potential of PCG. By embracing diverse algorithms and tailoring them to specific contexts, we can paint an ever-evolving canvas of game content with increasing vibrancy and depth.

9.2.3 Advanced Comparative Studies

Comparative studies using the most recent optimization algorithms offer a viable way to further our understanding of the advantages and disadvantages of particular algorithms, such as Genetic Algorithms (GA), particle swarm optimization (PSO), and Artificial Bee Colony (ABC). Using this approach allows us to explore the complexities of algorithmic performance in greater detail and identify important information that may have avoided the attention of previous studies.

This comprehensive strategy may apply advanced metaheuristic techniques that have not been previously investigated within the parameters of our study, including differential evolution, evolutionary strategies, or nature-inspired algorithms. These cutting-edge methods provide complex insights into optimization procedures and can show different ways to achieve ideal outcomes in Procedural Content Generation (PCG).

Furthermore, by comparing our study results with those obtained by applying cutting-edge algorithms, we can more effectively understand the relative effectiveness and possible drawbacks of each strategy. This comparative study strengthens the validity of our results and allows for a more sophisticated assessment of the al-

gorithmic performance in various PCG settings.

The knowledge gained from this comparative study has several important applications. In real world PCG applications, where the choice of the optimization approach can have a significant impact on the quality and diversity of the created material, they operate as a compass to guide algorithm selection. Through the application of comparative study results, practitioners can make well-informed judgements that are in line with the particular needs and limitations of their content creation endeavors.

In summary, conducting comparison studies that use cutting-edge optimization algorithms provides an opportunity to gain a deeper understanding and make better decisions in the field of PCG. We can open up new possibilities for innovation and optimization by carefully analyzing the data and carefully interpreting the findings. This will eventually propel the development of content creation strategies in the gaming industry and beyond.

9.2.4 Implications Beyond Gaming

Insights gained from studying metaheuristic algorithms in PCG for video games may extend beyond the gaming industry. These algorithms can potentially be applied to other domains that require optimization or content generation, such as art, music, architecture, and simulation.

Furthermore, this study emphasizes the importance of continued research and collaboration in the fields of metaheuristic algorithms and PCG. Further exploration of hybrid approaches, integration of domain-specific knowledge, and validation through real-world applications could advance the state-of-the-art content generation and algorithmic design.

Overall, the implications of studying metaheuristic algorithms in PCG extend beyond game development, with potential benefits for diverse industries and cre-

ative endeavors. By leveraging the power of metaheuristic optimization techniques, developers can unlock new possibilities for content creation, innovation, and user experience enhancement.

Additionally, although GA, PSO, and ABC are well-established metaheuristic algorithms, there is a vast landscape of alternative approaches that are yet to be explored in the context of PCG. Future research could investigate novel metaheuristic algorithms inspired by biological, social, or physical phenomena, such as cultural algorithms, firefly algorithms, and harmony search. By exploring new algorithmic paradigms, researchers can discover innovative solutions to challenging PCG problems that traditional algorithms cannot address.

9.2.5 The Role of Generative AI in Game Design and Meta-Heuristics

Generative Artificial Intelligence (Gen AI) is transforming game design and meta-heuristic optimization, driven by advancements in deep learning and generative models [184]. This technology enables unprecedented capabilities in procedural content generation, automated level design, and dynamic storytelling. By leveraging techniques such as Generative Adversarial Networks (GANs) and Transformer models, Gen AI automates content creation, enhances non-player character behavior, and facilitates real-time narrative adaptation. These advancements allow developers to shift their focus from manual design tasks to higher-level creative innovation, expanding the possibilities of interactive entertainment and pushing the boundaries of what games can achieve [185], [186].

Beyond game design, Gen AI is revolutionizing meta-heuristic optimization by introducing adaptive learning mechanisms that significantly enhance efficiency and problem-solving capabilities. The integration of reinforcement learning allows for dynamic adjustments in the parameters of evolutionary algorithms, accelerating convergence and improving solution quality. The ability of Gen AI to generate diverse search spaces and refine optimization strategies suggests a powerful synergy that could lead to more robust, scalable, and intelligent frameworks for solving complex computational challenges.

The future of Gen AI in game design and meta-heuristic optimization presents numerous exciting opportunities. Hybrid models that combine Gen AI with optimization techniques could lead to more efficient and creative approaches, such as using AI to initialize evolutionary algorithm populations or refine search processes. Empirical studies will be essential in evaluating the impact of Gen AI-augmented optimization strategies in real-world game development, particularly in assessing the quality and effectiveness of AI-generated content through player feedback. The potential of Gen AI to create adaptive gameplay and personalized experiences is another promising avenue, as AI-driven systems could dynamically adjust game mechanics based on player behavior and preferences, resulting in more engaging and immersive interactions. Further, advances in computational efficiency, including optimized training methods and distributed computing resources, will enable the large-scale deployment of these technologies in game development, making AI-driven design more accessible and efficient.

Gen AI is redefining how games are designed, optimized, and experienced, offering new ways to automate complex development processes and enhance creative possibilities. The continued exploration of hybrid AI models, adaptive game mechanics, and scalable computing solutions will unlock the next generation of intelligent game design and optimization strategies, shaping a future where AI-driven creativity and efficiency redefine interactive entertainment.

9.3 Practical Consideration

Currently, PCG is at the edge of a revolutionary phase [1] therefore, the intersection of theoretical developments and practical implementation has become critical. This section methodically covers practical problems and emerging opportunities that support the creation of engaging and diverse gaming experiences.

Choosing an algorithm that balances power requirements with resource limits is crucial. Simplified algorithms, such as PSO, may be useful for limited-resource projects, providing a reasonable balance between efficiency and performance. On the other hand, parallelization approaches provide up new possibilities for compu-

tationally demanding activities. This method has the potential to significantly accelerate content production for algorithms, such as PSO, which are naturally parallel. Furthermore, seamlessly integrating current game production workflows is crucial. Even though there are specialized libraries available for particular algorithms, looking into user-friendly choices can speed up the development cycle and facilitate seamless transfer from theory to practice.

Setting priorities for options with understandable parameters and mechanism interpretations is necessary to fine-tune the behavior of the algorithm. Greater control over created content is made possible by this transparency, which also makes it easier for users to have personalized experiences. Moving beyond popularity, we explore contemporary metaheuristic developments. Techniques, such as deep reinforcement learning and algorithms inspired by nature, have considerable potential. It is an exciting promise for future studies to investigate how these strategies might be applied to specific PCG difficulties and possibly even exceed the existing solutions.

Although computational efficiency is a key consideration, other limitations require further attention. Future studies should focus on hybridizing algorithms with other optimization methods or parallelization techniques. This collaborative effort has the potential to further reduce the time complexity and enhance the overall performance, unlocking a new level of efficiency.

Furthermore, caution must be exercised because of the possibility of bias in algorithms or training data. As these biases have the potential to produce unfair or discriminating results in the created material, it is crucial for responsible PCG development to design mitigation techniques and promote social and inclusive behavior. Finally, transparency is a need rather than a luxury. Building trust and understanding is facilitated by recording the decision-making process and offering justifications for the generated content, particularly when it significantly affects players. Everyone benefits from a more responsible and entertaining gaming experience as a result of our dedication to transparency.

Through careful attention to these pragmatic issues and the willingness to adapt to the opportunities awaiting them, researchers and developers can work together to reach the peak of metaheuristic-driven PCG. This joint endeavor to bring about a new era of content creation marked by efficiency, control, justice, and eventually the creation of genuinely immersive and captivating gaming experiences. There are many obstacles to a journey; however, there are also many opportunities for growth. This will open the door for procedural content generation to achieve unprecedented levels of originality, engagement, and diversity.

9.4 Limitations

This thesis acknowledges these limitations and seeks to maintain a forward-looking perspective by identifying areas for future research. Before discussing these limitations, it is important to acknowledge the following:

- **Although this study offers valuable insights into the application of metaheuristic algorithms in PCG for video games, it is crucial to acknowledge its limitations.**
- **The evaluation focused primarily on race track and map generation tasks, potentially limiting the generalizability of the findings to other PCG domains, such as character design or dialogue generation.**
- **The scope of this study necessarily restricted the exploration to a selection of metaheuristic algorithms, leaving room for further investigation of alternative approaches that might offer complementary strengths.**
- **The chosen evaluation metrics may not fully capture all nuances of the generated content quality, highlighting the need for further refinement and validation to ensure a comprehensive assessment in future research.**
- **Recognizing these limitations underscores the importance of continued exploration and the potential for further advancements in PCG through the thoughtful application of diverse metaheuristic approaches.**

While this study focuses on evaluating solution quality and convergence speed of PSO, ABC, and GA in PCG, it is important to acknowledge their broader impact on game development costs. Metaheuristic algorithms contribute to cost reduction by automating content generation, minimizing manual design efforts, and improving computational efficiency. Faster convergence leads to lower processing costs, while higher-quality solutions reduce the need for manual adjustments, ultimately saving development time. Although our research does not explicitly analyze cost savings, the efficiency of these algorithms indirectly influences overall development costs. Future studies could explore a comparative cost-benefit analysis to quantify these financial implications in PCG.

Metaheuristic algorithms often require fine-tuning of various parameters to achieve optimal performance. Understanding the impact of these parameters and their interactions can be complex and time consuming. Moreover, these algorithms may be highly sensitive to parameter settings, requiring extensive experimentation and expertise to determine suitable configurations.

The findings from our study focused primarily on race track and map layout generation tasks. However, it remains unclear how well these results generalize to other PCG domains such as character generation, texture synthesis, and level design. Further research should replicate our comparative analysis across a broader range of PCG applications, to assess the robustness and applicability of metaheuristic algorithms in diverse contexts.

Moreover, Metaheuristic algorithms, particularly those that involve large search spaces or complex fitness functions, can be computationally intensive. Generating high-quality game content using these algorithms may require significant computational resources and time, which could pose challenges for developers working under constraints such as project deadlines or hardware limitations.

One limitation of our study is the lack of consideration of player preferences and feedback when evaluating generated content. Future research could explore methods for integrating player feedback into the PCG process by leveraging tech-

niques such as interactive evolution, preference learning, and player modeling. By incorporating player preferences, developers can create personalized and engaging gaming experiences that are tailored to individual player preferences.

The findings from our study on metaheuristic algorithms in PCG may not always generalize well across different games, genres, or platforms. The effectiveness of a particular algorithm may depend on the specific characteristics of the game, such as the type of content generated, player preferences, or design constraints. Therefore, the conclusions drawn from a single study may not necessarily apply to all contexts.

Assessing the quality of the generated game content is inherently subjective and may vary depending on individual preferences, player experiences, and design objectives. While quantitative metrics, such as fitness scores or computational efficiency, can provide objective measures of algorithm performance, they may not comprehensively capture aspects of gameplay, immersion, or player satisfaction.

Focusing solely on performance metrics or specific PCG tasks may lead to overfitting or bias in algorithm evaluation. Algorithms optimized for a particular task may not be generalizable to new scenarios or real-world scenarios. Additionally, biases in the experimental design or result interpretation could have impacted the validity and reliability of the study findings.

The use of automated content generation techniques, including metaheuristic algorithms, raises ethical concerns regarding content originality, fairness, and inclusivity. Developers must ensure that generated content complies with legal regulations, respects intellectual property rights, and avoids perpetuating harmful stereotypes or biases.

As metaheuristic algorithms continue to play a significant role in shaping digital content creation, it is crucial to consider the ethical and societal implications of their use. Future research should explore topics, such as algorithmic bias, fairness,

transparency, and accountability in PCG systems. By addressing these issues, developers can ensure that PCG technologies are deployed responsibly and ethically, thereby promoting inclusivity and gaming diversity.

Effective application of metaheuristic algorithms in PCG requires collaboration among experts from diverse fields, including computer science, game design, psychology, and ethics. Bridging disciplinary gaps, effectively communicating complex concepts, and integrating domain-specific knowledge can be challenging, but essential, for successful algorithm development and deployment.

Advances in machine learning and artificial intelligence (AI) have provided exciting opportunities to enhance PCG capabilities. Future research could explore the integration of machine learning models, such as deep learning and reinforcement learning, into metaheuristic algorithms for PCG tasks. By leveraging AI techniques, developers can create adaptive, intelligent, and context-aware PCG systems capable of generating highly immersive and personalized gaming experiences.

Addressing these limitations requires careful consideration of the experimental design, methodological rigor, transparency in reporting, and ongoing dialogue among researchers, developers, and stakeholders. Despite these challenges, the study of metaheuristic algorithms in PCG offers valuable insights into and opportunities for innovation in game development.

9.5 Conclusions

In summary, this study establishes the groundwork for further investigations into optimization and metaheuristic algorithms. The ongoing improvement and development of these algorithms will help solve challenging real-world issues and expand the optimization field.

This study adds empirical support and understanding of the relative efficacy of metaheuristic algorithms in the developing field of PCG. Although Genetic Algo-

rithms have historically dominated the PCG industry, competing strategies such as ABC and PSO have the potential to broaden and improve the creation of game content. Game developers may create immersive and fascinating gaming experiences by utilizing a variety of metaheuristic strategies and embracing a deep understanding of algorithm performance.

The exploration of metaheuristic algorithms in PCG is a continuous process that offers opportunities for new discoveries, creative solutions, and cooperative effort. Metaheuristic algorithms are expected to be crucial in determining the direction of procedural content creation and interactive entertainment as the gaming industry develops further.

The flexibility and adaptability of metaheuristic algorithms provide game developers with an effective arsenal for creating compelling and varied game material. Metaheuristics such as Artificial Bee Colony (ABC), particle swarm optimization (PSO), and Genetic Algorithms (GA) offer creative answers to intricate optimization challenges, ranging from race tracks to map layouts. In the ever-changing world of game development, their aptitude for navigating complex solution spaces and striking the right balance between exploration and exploitation makes them priceless assets.

Nevertheless, there are certain restrictions on the success of metaheuristic algorithms in PCG. The necessity of meticulous experimentation, parameter adjustment, and interdisciplinary collaboration is highlighted by issues such as algorithm complexity, computer resource requirements, and subjectivity in evaluation measures. Furthermore, a major obstacle to the generalizability of the results across various games, genres, and platforms is the need for context-specific analyses and open reporting.

Despite these difficulties, researching metaheuristic algorithms in PCG provides insightful information on player experience, game design, and algorithm performance. The full potential of metaheuristics in influencing the direction of game con-

tent creation can be realised by academics and developers by resolving constraints, encouraging interdisciplinary collaboration, and embracing ethical considerations. In conclusion, even though metaheuristic algorithms may not be able to solve every PCG problem, further research and development could lead to advancements in both the science and art of game production. The gaming industry can harness the revolutionary potential of metaheuristic algorithms to produce immersive, diverse, and fascinating global gaming experiences for players by embracing innovation, critical inquiry, and responsible conduct.

In summary, the assessment and comparison of metaheuristic algorithms, specifically GA, PSO, and ABC, in game map generation underscored the strengths of ABC and PSO in terms of convergence speed and content quality. This study advocates the incorporation of diverse metaheuristic algorithms into PCG methodologies, opening avenues for elevating content generation and crafting captivating gaming experiences. These findings propel PCG forward and offer valuable insights for game developers seeking efficient and effective approaches for game content generation.

This thesis concludes by discussing opportunities, challenges, and future research directions in the field of PCG using metaheuristics. This highlights the need to explore alternative metaheuristic approaches, enhance the quality of fitness functions, integrate algorithms, promote diversity, consider the user experience, and establish evaluation metrics. Ethical considerations regarding PCG systems have also been identified as crucial aspects requiring further attention.

In summary, this comprehensive analysis provides valuable insights for researchers and game developers who intend to use optimization techniques to generate high-quality content in video games and other digital media. It expands the knowledge of how different algorithms are suited to PCG, lays the groundwork for further studies to determine the most suitable algorithms for games, and offers practical recommendations to inform the selection of these methods, ensuring that the resulting games have the potential to satisfy both the designer and player requirements.

References

[1] **Noor Shaker, Julian Togelius, and Mark J Nelson. Procedural content generation in games. *springer*, 2016.**

[2] **Julian Togelius, Noor Shaker, and Mark J Nelson. The search-based approach. *Computational Synthesis and Creative Systems, Cham, Springer*, pages 17--30, 2016.**

[3] **Nathan Sturtevant and Matheus Ota. Exhaustive and semi-exhaustive procedural content generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14, pages 109--115, 2018.**

[4] **Ashwin Ram, Santiago Ontañón, and Manish Mehta. Artificial intelligence for adaptive computer games. In *FLAIRS*, pages 22--29, 2007.**

[5] **Yuzhong Zhang, Guixuan Zhang, and Xinyuan Huang. A survey of procedural content generation for games. In *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, pages 186--190. IEEE, 2022.**

[6] **Georg Volkmar, Nikolas Mähl mann, and Rainer Malaka. Procedural content generation in competitive multiplayer platform games. In *Entertainment Computing and Serious Games: First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11--15, 2019, Proceedings 1*, pages 228--234. Springer, 2019.**

[7] **Noor Shaker, Georgios N Yannakakis, and Julian Togelius. Towards player-driven procedural content generation. In *Proceedings of the 9th conference on Computing Frontiers*, pages 237--240, 2012.**

[8] Keyuan Zhang, Jiayu Bai, and Jialin Liu. Generating game levels of diverse behaviour engagement. In *2022 IEEE Conference on Games (CoG)*, pages 167--174. IEEE, 2022.

[9] Alexander Gellel and Penny Sweetser. A hybrid approach to procedural generation of roguelike video game levels. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, pages 1--10, 2020.

[10] Michael Beukman, Christopher W Cleghorn, and Steven James. Procedural content generation using neuroevolution and novelty search for diverse video game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1028--1037, 2022.

[11] Matheus RF Mendonça and Artur Ziviani. Network-based procedural story generation. *Computers in Entertainment (CIE)*, 16(3):1--18, 2018.

[12] Hong Yu and Mark O Riedl. A sequential recommendation approach for interactive personalized story generation. In *AAMAS*, volume 12, pages 71--78, 2012.

[13] Tianhan Gao and Jiahui Zhu. A survey of procedural content generation of natural objects in games. In *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 275--279. IEEE, 2022.

[14] Nathaniel Finney and Jordi Janer. Soundscape generation for virtual environments using community-provided audio databases. In *W3C Workshop: Augmented Reality on the Web*. Barcelona Spain, 2010.

[15] Alba Amato. Procedural content generation in the game industry. *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, pages 15--25, 2017.

[16] Walaa Baghdadi, Fawzya Shams Eddin, Rawan Al-Omari, Zeina Alhalawani, Mohammad Shaker, and Noor Shaker. A procedural method for automatic generation of spelunky levels. In *European conference on the applications of evolutionary computation*, pages 305--317. Springer, 2015.

[17] Catherine Flick, L Meghan Dennis, and Andrew Reinhard. Exploring simulated game worlds: Ethics in the no man's sky archaeological survey. ORBIT, 2017.

[18] Jakob Schaal. Procedural terrain generation. a case study from the game industry. *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, pages 133--150, 2017.

[19] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1--22, 2013.

[20] David E Goldberg. Genetic protocols in search, optimization and machine learning, 1989.

[21] Miroslav Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *The Knowledge Engineering Review*, 13(4):409--412, 1999.

[22] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[23] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172--186, 2011.

[24] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1):19--37, 2021.

[25] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2018.

[26] Nicholas J Radcliffe and Patrick D Surry. Fundamental limitations on search algorithms: Evolutionary computing in perspective. *Computer Science Today: Recent Trends and Developments*, pages 275--291, 2005.

[27] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533--549, 1986.

[28] Rohit K Sachan and Dharmender S Kushwaha. Inspirations from nature for meta-heuristic algorithms: A survey. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 14(6):1706--1718, 2021.

[29] WK Wong and Chew Ing Ming. A review on metaheuristic algorithms: recent trends, benchmarking and applications. In *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, pages 1--5. IEEE, 2019.

[30] Leonardo Tortoro Pereira, Paulo Victor de Souza Prado, Rafael Miranda Lopes, and Claudio Fabiano Motta Toledo. Procedural generation of dungeons' maps and locked-door missions through an evolutionary algorithm validated with players. *Expert Systems with Applications*, 180:115009, 2021.

[31] Breno MF Viana, Leonardo T Pereira, Claudio FM Toledo, Selan R dos Santos, and Silvia MDM Maia. Feasible--infeasible two-population genetic algorithm to evolve dungeon levels with dependencies in barrier mechanics. *Applied Soft Computing*, 119:108586, 2022.

[32] Adam Slowik and Halina Kwasnicka. Nature inspired methods and their industry applications—swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, 14(3):1004--1015, 2017.

[33] Amit Garg, Pawan Gill, Parveen Rathi, KK Garg, et al. An insight into swarm intelligence. *International Journal of Recent Trends in Engineering*, 2(8):42, 2009.

[34] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing: From basics to applications. *Handbook of metaheuristics*, pages 1--35, 2019.

[35] Adeel Zafar, Hasan Mujtaba, and Mirza Omer Beg. Search-based procedural content generation for gvg-lg. *Applied Soft Computing*, 86:105909, 2020.

[36] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pages 39--43. Ieee, 1995.

[37] **Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial intelligence review*, 42:21--57, 2014.**

[38] **Andy M Connor, Taura J Greig, and Jan Kruse. Evaluating the impact of procedurally generated content on game immersion. *The Computer Games Journal*, 6:209--225, 2017.**

[39] **Hamed Namdar. Developing an improved approach to solving a new gas lift optimization problem. *Journal of Petroleum Exploration and Production Technology*, 9(4):2965--2978, 2019.**

[40] **S Venkata Lakshmi and Valli Kumari Vatsavayi. Query optimization using clustering and genetic algorithm for distributed databases. In *2016 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1--8. IEEE, 2016.**

[41] **AM Connor, TJ Greig, and J Kruse. Evolutionary generation of game levels. *XXX*, 2018.**

[42] **Christopher Simons and James Smith. A comparison of evolutionary algorithms and ant colony optimization for interactive software design. In *Proceedings of the 4th Symposium on Search Based-Software Engineering*, page 37. Springer Berlin, 2012.**

[43] **Nicholas Muir and Steven James. Combining evolutionary search with behaviour cloning for procedurally generated content. *arXiv preprint arXiv:2207.14772*, 2022.**

[44] **Andreas Scheibenpflug, Johannes Karder, Susanne Schaller, Stefan Wagner, and Michael Affenzeller. Evolutionary procedural 2d map generation using novelty search. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 39--40, 2016.**

[45] **Laura Calvet Liñán, Ángel Alejandro Juan Pérez, Carles Serrat Piè, and Jana Ries. A statistical learning based approach for parameter fine-tuning of metaheuristics. *SORT: statistics and operations research transactions*, 40(1):201--224, 2016.**

[46] **Anupriya Gogna and Akash Tayal. Comparative analysis of evolutionary algorithms for image enhancement. *International Journal of Metaheuristics*, 2(1):80--100, 2012.**

[47] **Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*, pages 1--8. IEEE, 2019.**

[48] **Georgios N Yannakakis and Julian Togelius. Experience-driven procedural content generation. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 519--525. IEEE, 2015.**

[49] **Martín González-Hermida, Enrique Costa-Montenegro, Beatriz Legerén-Lago, and Antonio Pena-Giménez. Study of artificial intelligent algorithms applied in procedural content generation in video games. *Eludamos: Journal for Computer Game Culture*, 10(1):39--54, 2019.**

[50] **Sana Alyaseri, Roopak Sinha, and Parma Nand. Systematic literature review of meta-heuristic algorithms and their application in procedural content generation in the context of computer games. Submitted to Swarm and Evolutionary Computation Journal for publication, 2023.**

[51] **Parampreet Kaur, Jill Stoltzfus, and Vikas Yellapu. Descriptive statistics. *International Journal of Academic Medicine*, 4(1):60--63, 2018.**

[52] **M Lakshman, Leena Sinha, Moumita Biswas, Maryann Charles, and NK Arora. Quantitative vs qualitative research methods. *The Indian Journal of Pediatrics*, 67:369--377, 2000.**

[53] **Andy M Connor and Kristina Shea. A comparison of semi-deterministic and stochastic search techniques. In *Evolutionary Design and Manufacture: Selected Papers from ACDM'00*, pages 287--298. Springer, 2000.**

[54] **Krishna Gopal Dhal, Arunita Das, Jorge Gálvez, Swarnajit Ray, and Sanjoy Das. An overview on nature-inspired optimization algorithms and their possible application in image processing domain. *Pattern Recognition and Image Analysis*, 30:614--631, 2020.**

[55] **Anupriya Gogna and Akash Tayal. Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4):503--526, 2013.**

[56] **Jakub Kowalski, Radosław Miernik, Piotr Pytlik, Maciej Pawlikowski, Krzysztof Piecuch, and Jakub Skekowski. Strategic features and terrain generation for balanced heroes of might and magic iii maps. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1--8. IEEE, 2018.**

[57] **Umberto Picariello, Daniele Loiacono, Fabio Mosca, and Pier Luca Lanzi. A framework to create collaborative games for team building using procedural content generation. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2365--2372. IEEE, 2020.**

[58] **Oliver Korn, Michael Blatz, Adrian Rees, Jakob Schaal, Valentin Schwind, and Daniel Görlich. Procedural content generation for game props? a study on the effects on user experience. *Computers in Entertainment (CIE)*, 15(2):1--15, 2017.**

[59] **Rong Zhang et al. Automatic generation of real-time animation game learning levels based on artificial intelligence assistant. *Scientific Programming*, 2022, 2022.**

[60] **Zulfiquar N Ansari and Sachin D Daxini. A state-of-the-art review on meta-heuristics application in remanufacturing. *Archives of Computational Methods in Engineering*, 29(1):427--470, 2022.**

[61] **Vivek Sharma and Ashish Kumar Tripathi. A systematic review of meta-heuristic algorithms in iot based application. *Array*, 14:100164, 2022.**

[62] **Ali R Kashani, Charles V Camp, Mehdi Rostamian, Koorosh Azizi, and Amir H Gandomi. Population-based optimization in structural engineering: a review. *Artificial Intelligence Review*, pages 1--108, 2022.**

[63] **Shahab Wahhab Kareem, Kurdistan Wns Hama Ali, Shavan Askar, Farah Sami Xoshaba, and Roojwan Hawezi. Metaheuristic algorithms in optimization and its application: A review. *JAREE (Journal on Advanced Research in Electrical Engineering)*, 6(1), 2022.**

[64] **Yu Zhu and Lin Wu. Structure study of multiple traveling salesman problem using genetic algorithm. In *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 323--328. IEEE, 2019.**

[65] **Absalom E Ezugwu, Amit K Shukla, Rahul Nath, Andronicus A Akinyelu, Jeffery O Agushaka, Haruna Chiroma, and Pranab K Muhuri. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, 54:4237--4316, 2021.**

[66] **Chimmiri Venkateswarlu. A metaheuristic tabu search optimization algorithm: Applications to chemical and environmental processes. In *Engineering Problems-Uncertainties, Constraints and Optimization Techniques*, pages 245--277. IntechOpen, 2021.**

[67] **Yi Liu and Shengqing Tang. 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)*, pages 923--929. IEEE, 2017.**

[68] **Lei Wang, Xin Zhang, and Xiu Zhang. Antenna array design by artificial bee colony algorithm with similarity induced search method. *IEEE Transactions on Magnetics*, 55(6):1--4, 2019.**

[69] **A Pal, FTS Chan, B Mahanty, and MK Tiwari. 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, 2011.**

[70] **Mohamed K Omar and Kumaran Ramakrishnan. Solving non-oriented two dimensional bin packing problem using evolutionary particle swarm optimisation. *International Journal of Production Research*, 51(20):6002--6016, 2013.**

[71] **RD Villarroel, DF García, MA Dávila, and EF Caicedo. Particle swarm optimization vs genetic algorithm, application and comparison to determine the moisture diffusion coefficients of pressboard transformer insulation. *IEEE Transactions on Dielectrics and Electrical Insulation*, 22(6):3574--3581, 2015.**

[72] Jui-Sheng Chou and Asmare Molla. Recent advances in use of bio-inspired jellyfish search algorithm for solving optimization problems. *Scientific Reports*, 12(1):19157, 2022.

[73] Barbara Kitchenham, Stuart Charters, et al. Guidelines for performing systematic literature reviews in software engineering, 2007.

[74] Daniel Ashlock, Colin Lee, and Cameron McGuinness. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260--273, 2011.

[75] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. Automatic track generation for high-end racing games using evolutionary computation. *IEEE Transactions on computational intelligence and AI in games*, 3(3):245--259, 2011.

[76] Fausto Mourato, Manuel Próspero dos Santos, and Fernando Birra. Automatic level generation for platform videogames using genetic algorithms. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, pages 1--8, 2011.

[77] Nathan Sorenson, Philippe Pasquier, and Steve DiPaola. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):229--244, 2011.

[78] Cameron McGuinness and Daniel Ashlock. Decomposing the level generation problem with tiles. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 849--856. IEEE, 2011.

[79] Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. Procedural generation of quests for games using genetic algorithms and automated planning. In *SBGames*, pages 144--153, 2019.

[80] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Sentient sketchbook: computer-assisted game level authoring. *ACM*, 2013.

[81] Lucas Ferreira, Leonardo Pereira, and Claudio Toledo. A multi-population genetic algorithm for procedural generation of levels for platform games. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 45--46, 2014.

[82] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N Yannakakis, and Corrado Grappiolo. Controllable procedural map generation via multiobjective evolution. *Genetic Programming and Evolvable Machines*, 14:245--277, 2013.

[83] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5899--5908, 2018.

[84] Carl Hultquist, James Gain, and David Cairns. An adjectival interface for procedural content generation. In *Intelligent Computer Graphics 2009*, pages 143--165. Springer, 2009.

[85] Wen Xia and Bhojan Anand. Game balancing with ecosystem mechanism. In *2016 international conference on data mining and advanced computing (SAPIENCE)*, pages 317--324. IEEE, 2016.

[86] Luiz Jonatã Pires de Araújo, Alexandr Grichshenko, Rodrigo Lankaites Pinheiro, Rommel D Saraiva, and Susanna Gimaeva. Map generation and balance in the terra mystica board game using particle swarm and local search. In *Advances in Swarm Intelligence: 11th International Conference, ICSI 2020, Belgrade, Serbia, July 14--20, 2020, Proceedings 11*, pages 163--175. Springer, 2020.

[87] Rafael Guerra De Pontes, Herman Martins Gomes, and Igor Santa Ritta Seabra. Particle swarm optimization for procedural content generation in an endless platform game. *Entertainment Computing*, 43:100496, 2022.

[88] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Constrained novelty search: A study on game content generation. *Evolutionary computation*, 23(1):101--129, 2015.

[89] **Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*, pages 1--8. IEEE, 2019.**

[90] **Alexandre Santos Melotti and Carlos Henrique Valerio de Moraes. Evolving roguelike dungeons with deluged novelty search local competition. *IEEE Transactions on Games*, 11(2):173--182, 2018.**

[91] **Arash Moradi Karkaj and Shahriar Lotfi. Using estimation of distribution algorithm for procedural content generation in video games. *Genetic Programming and Evolvable Machines*, 23(4):495--533, 2022.**

[92] **Breno MF Viana, Leonardo T Pereira, and Claudio FM Toledo. Illuminating the space of enemies through map-elites. In *2022 IEEE Conference on Games (CoG)*, pages 17--24. IEEE, 2022.**

[93] **Febri Abdullah, Pujana Paliyawan, Ruck Thawonmas, and Fitra A Bachtiar. Generating angry birds-like levels with domino effects using constrained novelty search. In *2020 IEEE Conference on Games (CoG)*, pages 698--701. IEEE, 2020.**

[94] **Nicolas A Barriga. A short introduction to procedural content generation algorithms for videogames. *International Journal on Artificial Intelligence Tools*, 28(02):1930001, 2019.**

[95] **Daniel Blasco, Jaime Font, Francisca Pérez, and Carlos Cetina. Procedural content improvement of game bosses with an evolutionary algorithm. *Multimedia Tools and Applications*, 82(7):10277--10309, 2023.**

[96] **Eirik Høgdahl Skjærseth, Harald Vinje, and Ole Jakob Mengshoel. Novelty search for evolving interesting character mechanics for a two-player video game. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 321--322, 2021.**

[97] **Hongqiao Zhang, Xiaohua Zeng, and Lingfei Duan. Procedural content generation for room maze. In *2018 3rd Joint International Information Technology, Mechanical and Electronic Engineering Conference (JIMEC 2018)*, pages 265--269. Atlantis Press, 2018.**

[98] Andy M Connor and Derek G Tilley. A comparison of two methods applied to the optimisation of fluid power circuits. *arXiv preprint arXiv:1605.03667*, 2016.

[99] Madhumita Panda. Performance comparison of genetic algorithm, particle swarm optimization and simulated annealing applied to tsp. *International Journal of Applied Engineering Research*, 13(9):6808--6816, 2018.

[100] Frederick Schmidt, Stephen MacDonell, and Andy M Connor. Multi-objective reconstruction of software architecture. *International Journal of Software Engineering and Knowledge Engineering*, 28(06):869--892, 2018.

[101] Habib Youssef, Sadiq M Sait, and Hakim Adiche. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, 14(2):167--181, 2001.

[102] Micael Couceiro, Pedram Ghamisi, Micael Couceiro, and Pedram Ghamisi. *Particle swarm optimization*. Springer, 2016.

[103] Fahad S Abu-Mouti and Mohamed E El-Hawary. Overview of artificial bee colony (abc) algorithm and its applications. In *2012 IEEE International Systems Conference SysCon 2012*, pages 1--6. IEEE, 2012.

[104] Georgios N Yannakakis, Julian Togelius, Georgios N Yannakakis, and Julian Togelius. Generating content. *Artificial intelligence and games*, pages 151--202, 2018.

[105] Arman Balali Moghadam and Marjan Kuchaki Rafsanjani. A genetic approach in procedural content generation for platformer games level creation. In *2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 141--146. IEEE, 2017.

[106] Leonardo T Pereira and Claudio FM Toledo. Speeding up search-based algorithms for level generation in physics-based puzzle games. *International Journal on Artificial Intelligence Tools*, 26(05):1760019, 2017.

[107] Ji-Min Kim, Sun-Jeong Kim, and Seokmin Hong. Players adaptive monster generation technique using genetic algorithm. *Journal of Internet Computing and Services*, 18(2):43--51, 2017.

[108] **Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. Procedural generation of branching quests for games. *Entertainment Computing*, 43:100491, 2022.**

[109] **Tansel Dokeroglu, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040, 2019.**

[110] **Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, and Georgios N Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 265--272. IEEE, 2010.**

[111] **Daniel Michelon De Carli, Fernando Bevilacqua, Cesar Tadeu Pozzer, and Marcos Cordeiro d'Ornellas. A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian symposium on games and digital entertainment*, pages 26--35. IEEE, 2011.**

[112] **Paul Hyunjin Kim and Roger Crawfis. The quest for the perfect perfect-maze. In *2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, pages 65--72. IEEE, 2015.**

[113] **Iztok Fister Jr, Matjaž Perc, Karin Ljubič, Salahuddin M Kamal, Andres Iglesias, and Iztok Fister. Particle swarm optimization for automatic creation of complex graphic characters. *Chaos, Solitons & Fractals*, 73:29--35, 2015.**

[114] **Alexandr Grichshenko, Luiz Jonatã Pires de Araújo, Susanna Gimaeva, and Joseph Alexander Brown. Using tabu search algorithm for map generation in the terra mystica tabletop game. In *Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, pages 119--122, 2020.**

[115] **Hafizh Adi Prasetya and Nur Ulfa Maulidevi. Search-based procedural content generation for race tracks in video games. *International Journal on Electrical Engineering & Informatics*, 8(4), 2016.**

[116] **Y Collette, P Siarry, and H-l Wong. 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, 2000.

[117] AM Leite Da Silva, LS Rezende, LM Honório, and LAF Manso. Performance comparison of metaheuristics to solve the multi-stage transmission expansion planning problem. *IET generation, transmission & distribution*, 5(3):360--367, 2011.

[118] Shubham Gupta, Hammoudi Abderazek, Betül Sultan Yıldız, Ali Riza Yildiz, Seyedali Mirjalili, and Sadiq M Sait. Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. *Expert Systems with Applications*, 183:115351, 2021.

[119] S Kannan, S Mary Raja Slochanal, and Narayana Prasad Padhy. Application and comparison of metaheuristic techniques to generation expansion planning problem. *IEEE transactions on Power Systems*, 20(1):466--475, 2005.

[120] Pedro Reche-López, Nicolas Ruiz-Reyes, S García Galán, and Francisco Jurado. Comparison of metaheuristic techniques to determine optimal placement of biomass power plants. *Energy conversion and management*, 50(8):2020--2028, 2009.

[121] Iztok Fister Jr, Xin-She Yang, Iztok Fister, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

[122] Shweta Dalwani and Anuradha Agarwal. Review on classification of nature inspired approach. *International Journal of Computer & Mathematical Sciences, IJCMS, ISSN 2347*, 8527, 2018.

[123] Oliver Kramer and Oliver Kramer. *Genetic algorithms*. Springer, 2017.

[124] Andrew Miles Connor. *The synthesis of hybrid mechanisms using genetic algorithms*. Liverpool John Moores University (United Kingdom), 1996.

[125] A Ghaheri, S Shoar, M Naderan, and SS Hoseini. The applications of genetic algorithms in medicine, oman med. *J*, 30(6):406--416, 2015.

[126] **Kim-Fung Man, Kit-Sang Tang, and Sam Kwong. Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics*, 43(5):519--534, 1996.**

[127] **Timmy Manning, Roy D Sleator, and Paul Walsh. Naturally selecting solutions: the use of genetic algorithms in bioinformatics. *Bioengineered*, 4(5):266--278, 2013.**

[128] **Wojciech Paszkowicz. 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, 2013.**

[129] **Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:337--370, 1996.**

[130] **Qing Wang, Pieter Spronck, and Rudolf Tracht. 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)*, volume 3, pages 1651--1656. IEEE, 2003.**

[131] **Tiago Alves, Jorge Coelho, and Luís Nogueira. 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*, pages 37--49. Springer, 2018.**

[132] **Vid Kraner, Iztok Fister Jr, and Lucija Brezočnik. Procedural content generation of custom tower defense game using genetic algorithms. In *International Conference "New Technologies, Development and Applications"*, pages 493--503. Springer, 2021.**

[133] **Sana Alyaseri and Alaa Aljanaby. Population based heuristic approaches for grid job scheduling. *International Journal of Computer Applications*, 91(5), 2014.**

[134] **Liming Zhang, Saisai Wang, Kai Zhang, Xiuqing Zhang, Zhixue Sun, Hao Zhang, Miguel Tome Chipecane, and Jun Yao. Cooperative artificial bee colony algorithm with multiple populations for interval multiobjective optimization problems. *IEEE Transactions on Fuzzy Systems*, 27(5):1052--1065, 2018.**

[135] Carlos Mora, Sandra Jardim, and Jorge Valente. Flora generation and evolution algorithm for virtual environments. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1--6. IEEE, 2021.

[136] Furkan Yücel and ELİF Sürer. 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, 2020.

[137] Chinmaya Sahu, Priyadarshi Biplab Kumar, and Dayal R Parhi. An intelligent path planning approach for humanoid robots using adaptive particle swarm optimization. *International Journal on Artificial Intelligence Tools*, 27(05):1850015, 2018.

[138] Qiang Zheng, Bai-Wei Feng, Zu-Yuan Liu, and Hai-Chao Chang. Application of improved particle swarm optimisation algorithm in hull form optimisation. *Journal of Marine Science and Engineering*, 9(9):955, 2021.

[139] Vaishali R Kulkarni and Veena Desai. Abc and pso: A comparative analysis. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1--7. IEEE, 2016.

[140] Željko Kanović, Vladimir Bugarski, and Todor Bačkalić. Ship lock control system optimization using ga, pso and abc: a comparative review. *Promet-Traffic&Transportation*, 26(1):23--31, 2014.

[141] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, 214(1):108--132, 2009.

[142] Matthew Settles, Brandon Rodebaugh, and Terence Soule. Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In *Genetic and Evolutionary Computation Conference*, pages 148--149. Springer, 2003.

[143] Amin Farahbakhsh and Amir Saman Kheirkhah. 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, 2023.

[144] Gerald Farin. Class a bézier curves. *Computer aided geometric design*, 23(7):573--581, 2006.

[145] Maxime Heckel. Cubic bezier: From math to motion, 2021.

[146] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091--8126, 2021.

[147] Bilge Kagan Dedeturk, Bahriye Akay, and Dervis Karaboga. Artificial bee colony algorithm and its application to content filtering in digital communication. *Nature-Inspired Metaheuristic Algorithms for Engineering Optimization Applications*, pages 337--355, 2021.

[148] Carlos A Souza Lima Jr, Celso Marcelo F Lapa, Cláudio Márcio do NA Pereira, João J da Cunha, and Antonio Carlos M Alvim. 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, 2011.

[149] Norfadzlan Yusup, Azlan Mohd Zain, and Siti Zaiton Mohd Hashim. Overview of pso for optimizing process parameters of machining. *Procedia Engineering*, 29:914--923, 2012.

[150] Tareq M Shami, Ayman A El-Saleh, Mohammed Alswaitti, Qasem Al-Tashi, Mhd Amen Summakieh, and Seyedali Mirjalili. Particle swarm optimization: A comprehensive survey. *IEEE Access*, 10:10031--10061, 2022.

[151] Sana Alyaseri and Andy Connor. Comparative analysis of metaheuristic algorithms for procedural race track generation in games. accepted for publication in International Journal of Applied Metaheuristic Computing (IJAMC), 2024.

[152] Daniel Karavolos, Antonios Liapis, and Georgios Yannakakis. A multifaceted surrogate model for search-based procedural content generation. *IEEE Transactions on Games*, 13(1):11--22, 2019.

[153] Marko Gulić, Martina Žuškin, and Vilim Kvaternik. An overview and comparison of selected state-of-the-art algorithms inspired by nature. *TEM Journal*, 12(3):1281, 2023.

[154] Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D Gaina, Julian Togelius, and Simon M Lucas. General video game ai: A multitrack framework

for evaluating agents, games, and content generation algorithms. *IEEE Transactions on Games*, 11(3):195--214, 2019.

[155] P Sam Daniel and Aroma G Sam. *Research methodology*. Gyan Publishing House, 2011.

[156] Julian Togelius, Mike Preuss, and Georgios N Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 workshop on procedural content generation in games*, pages 1--8, 2010.

[157] Hokey Min. Genetic algorithm for supply chain modelling: basic concepts and applications. *International Journal of Services and Operations Management*, 22(2):143--164, 2015.

[158] Hui Zhi and Sanyang Liu. Gray image segmentation based on fuzzy c-means and artificial bee colony optimization. *Journal of Intelligent & Fuzzy Systems*, 38(4):3647--3655, 2020.

[159] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft computing*, 22:387--408, 2018.

[160] Satyobroto Talukder. Mathematicle modelling and applications of particle swarm optimization, 2011.

[161] Ovat Friday Aje and Anyandi Adie Josephat. The particle swarm optimization (pso) algorithm application--a review. *Global Journal of Engineering and Technology Advances*, 3(3):001--006, 2020.

[162] Mohammed A El-Shorbagy and Aboul Ella Hassanien. Particle swarm optimization from theory to applications. *International Journal of Rough Sets and Data Analysis (IJRSDA)*, 5(2):1--24, 2018.

[163] Jiashan Zhang, Zhenzhu Zhang, and Xiaoqun Lin. An improved artificial bee colony with self-adaptive strategies and application. In *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pages 101--104. IEEE, 2021.

[164] Shiv Kumar Agarwal and Surendra Yadav. A comprehensive survey on artificial bee colony algorithm as a frontier in swarm intelligence. *Ambient Communications and Computer Systems: RACCCS-2018*, pages 125--134, 2019.

[165] **Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428--436, 2020.**

[166] **Ronald Palandi Cardoso, José Salvador da Motta Reis, Dayana Elizabeth Werderits Silva, Maria da Glória Diniz de Almeida, José Glenio Medeiros de Barros, and Nilo Antonio de Souza Sampaio. Scientific research trends about metaheuristics in process optimization and case study using the desirability function. *Revista de Gestão e Secretariado (Management and Administrative Professional Review)*, 14(3):3348--3367, 2023.**

[167] **Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, 52:2191--2233, 2019.**

[168] **Kamal Z Zamli. Enhancing generality of meta-heuristic algorithms through adaptive selection and hybridization. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 67--71. IEEE, 2018.**

[169] **Katherine M Malan and Andries P Engelbrecht. Fitness landscape analysis for metaheuristic performance prediction. In *Recent advances in the theory and application of fitness landscapes*, pages 103--132. Springer, 2014.**

[170] **Sana Alyaseri, Andy Connor, and Roopak Sinha. Exploring metaheuristic algorithms for enhanced game map generation in procedural content generation. Submitted for publication, 2024.**

[171] **Murat Ince. Bilstm and dynamic fuzzy ahp-ga method for procedural game level generation. *Neural Computing and Applications*, 33(15):9761--9773, 2021.**

[172] **Sasha Azad, Carl Saldanha, Cheng-Hann Gan, and Mark Riedl. Mixed reality meets procedural content generation in video games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 12, pages 22--26, 2016.**

[173] **Wei-Bing Liu and Xian-Jia Wang. An evolutionary game based particle swarm optimization algorithm. *Journal of computational and Applied Mathematics*, 214(1):30--35, 2008.**

[174] Jiaping Yang and Chee Kiong Soh. Structural optimization by genetic algorithms with tournament selection. *Journal of computing in civil engineering*, 11(3):195--200, 1997.

[175] Carlos A Coello Coello and Efrén Mezura-Montes. Handling constraints in genetic algorithms using dominance-based tournaments. In *Adaptive computing in design and manufacture V*, pages 273--284. Springer, 2002.

[176] Jeannie Fitzgerald and Conor Ryan. Selection bias and generalisation error in genetic programming. In *Sixth International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN2014*, 2014.

[177] AE Eiben, Martijn C Schut, and AR De Wilde. Boosting genetic algorithms with self-adaptive selection. In *2006 IEEE International Conference on Evolutionary Computation*, pages 477--482. IEEE, 2006.

[178] Diogo Freitas, Luiz Guerreiro Lopes, and Fernando Morgado-Dias. Particle swarm optimisation: a historical review up to the current developments. *Entropy*, 22(3):362, 2020.

[179] Anupam Biswas, Bhaskar Biswas, Anoj Kumar, and KK Mishra. Particle swarm optimisation with time varying cognitive avoidance component. *International Journal of Computational Science and Engineering*, 16(1):27--41, 2018.

[180] Ryan Roussel, Juan Pablo Gonzalez-Aguilera, Young-Kee Kim, Eric Wisniewski, Wanming Liu, Philippe Piot, John Power, Adi Hanuka, and Auralee Edelen. Turn-key constrained parameter space exploration for particle accelerators using bayesian active learning. *Nature communications*, 12(1):5612, 2021.

[181] Rodrigo Cupertino Bernardes, Lorena Lisbetd Botina, Renan dos Santos Araujo, Raul Narciso Carvalho Guedes, Gustavo Ferreira Martins, and Maria Augusta Pereira Lima. Artificial intelligence-aided meta-analysis of toxicological assessment of agrochemicals in bees. *Frontiers in Ecology and Evolution*, 10, 2022.

[182] Tarun Maini, RK Misra, and D Singh. Optimal feature selection using elitist genetic algorithm. In *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, pages 1--5. IEEE, 2015.

[183] Oliver Withington and Laurissa Tokarchuk. Compressing and comparing the generative spaces of procedural content generators. In *2022 IEEE Conference on Games (CoG)*, pages 143--150. IEEE, 2022.

[184] Mauro Cazzaniga, Ms Florence Jaumotte, Longji Li, Mr Giovanni Melina, Augustus J Panton, Carlo Pizzinelli, Emma J Rockall, and Ms Marina Mendes Tavares. *Gen-AI: Artificial intelligence and the future of work*. International Monetary Fund, 2024.

[185] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. Adversarial reinforcement learning for procedural content generation. In *2021 IEEE Conference on Games (CoG)*, pages 1--8. IEEE, 2021.

[186] Matthew C Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K Hoover, and Stefanos Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5922--5930, 2021.

[187] Richard Stangl. Procedural content generation: techniques and applications. In *ACM Proceedings*, 2017.

Appendices

Appendix A

Prelude - Manuscript 4

This study challenges the idea that a single optimal algorithm exists for PCG. This emphasises the crucial role of task-specific considerations in determining the most appropriate algorithm for a given task. Through a novel comparative analysis, we demonstrate the importance of contextual superiority: PSO emerges as highly effective in generating race tracks, whereas ABC demonstrates superior performance in map generation. Depending on the time constraints, the rapid exploration capability of PSO is advantageous for quick results, whereas the detailed search ability of ABC is preferable for achieving higher quality outcomes over a longer duration. This study extends beyond the traditional focus on GAs by highlighting the unique strengths of PSO and ABC. It advocates broader exploration beyond the confines of GA-centric approaches to PCG. We propose an approach termed ``Context-Aware Approach'' for algorithm selection, which underscores the importance of analyzing the complexity of the task, quality requirements, and desired solution speed. By adopting this context-aware approach and comprehending the interactions between task characteristics and algorithm capabilities, researchers and developers can make well-informed decisions, ultimately leading to the creation of engaging and personalized PCG experiences.

Appendix B

A Context Aware Approach Framework to Algorithm Selection for Search Based Procedural Content Generation (Manuscript 4)

B.1 Introduction

Procedural Content Generation (PCG) exploits algorithms and mathematical principles to automate in-game element creation encompassing procedural level, narrative, asset, and sound generation, thus uniquely enriching player experiences. In addition to improving the development processes and reducing expenses, PCG stimulate innovation and enhance player engagement. Over the past decade, PCG research has flourished by exploring diverse methods and types of content [1].

One prominent approach is search-based PCG, which utilizes metaheuristic algorithms, such as the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC). These algorithms iteratively refine content creation and offer developers flexibility and efficiency. However, understanding how these algorithms interact with different PCG tasks remains a challenge [187]. Our goal is to invite researchers to study more in this area by introducing these algorithms and to have more discussions on metaherstics in PCG.

This study extends the findings from two of our prior investigations, detailed in Study#1 [151] and Study#2 [170] to analyze the performance of GA, PSO, and ABC on two distinct PCG tasks: race track generation and map generation. These studies employed accurate methodologies to ensure a robust comparison, including controlling variables such as shared variables between algorithms (population size and iteration count) to isolate the impact of each algorithm on performance, addressing convergence criteria by running each algorithm 10 times with consistent parameter settings, and using statistical evaluation techniques such as best, mean, worst values, and standard deviations to achieve a comprehensive picture of algorithmic performance.

By leveraging these established methodologies and focusing on the results of previous studies, we aimed to identify generalizable patterns and insights applicable to broader PCG research. This will ultimately improve our understanding of PCG algorithms and their applications, thereby guiding future research and innovation in

this field.

The remainder of this paper is structured as follows in order to provide a more comprehensive understanding. Section B.2 examines the background and concepts of PCG, focusing on search-based approaches. Section B.3 presents the methodology. Section B.4 is the core of our investigation and presents a comparison of the performances of the three algorithms on selected PCG tasks. Section B.5 discusses the results, identifies generalizable patterns, and provides valuable insights into our PCG research. Finlay, Section B.6 summarizes the study's conclusions.

B.2 Related work and Background

PCG encompasses various techniques and algorithms to automate game content creation and enhance the player experience. One powerful method, search-based PCG, employs metaheuristic algorithms, such as GA, PSO, and ABC, for iterative refinement. However, evaluating the quality of the generated content remains a significant challenge, as noted by Togelius (2011) in the search-based PCG domain [23].

Metaheuristic approaches draw inspiration from both natural and computational processes, and provide efficient solutions for intricate problems in which exact solutions pose challenges. These versatile algorithms find applications across diverse domains such as optimization, machine learning, and PCG. Metaheuristic algorithms fall under the umbrella of nature-inspired approaches (NIAs) and replicate natural behaviors to solve optimization problems.

In the PCG domain, metaheuristic algorithms play a pivotal role in automating the creation of engaging content for video games. A prime example is the application of Evolutionary Algorithms, with GAs as a notable example. These algorithms mimic inherited behaviors, starting with a randomized population and evolving over several generations. GAs have demonstrated success in computer game development, particularly in enhancing Dungeon game levels through concept-based map generation [30], [31], [1].

Our studies #1 and #2 chose the GA, PSO, and ABC for comparison. The first algorithm is the GA [20] which stands out as a renowned evolutionary algorithm, drawing inspiration from natural selection processes. It initiates with random generation of an initial population, assesses the objective function for each individual, and results in a binary vector. Crossover involves creating new individuals by merging portions of the chromosomes from two distinct individuals, while mutation alters a randomly selected gene within an individual. The next generation comprises individuals with the best objective function values, and has found widespread application in PCG, particularly in domains such as level design for video games [35]. Its ability to explore a vast search space and discover optimal or near-optimal solutions makes it well suited for creating diverse and engaging game levels.

However, PSO [36] draws inspiration from the collective behavior of bird flocks in nature and represents a classic metaheuristic algorithm rooted in swarm intelligence. Unlike traditional populations, PSO focuses on generating location and velocity vectors for individuals at the initial stage. During each iteration, PSO identifies the best-known position for each particle and the overall swarm by utilizing this information to calculate new particle velocity values. Consequently, the optimal state of the swarm serves as a solution to the optimization problem. In the context of PCG, PSO has demonstrated effectiveness in tasks, such as race track [151]. Its ability to balance exploration and exploitation within a solution space makes it well suited for crafting diverse and playable game environments.

The final approach was ABC [37]. It is a metaheuristic algorithm inspired by the foraging behavior of honeybees. In ABC, the optimization process simulates the activities of the employed bees, onlookers, and scouts to discover optimal solutions. Employed bees exploit food sources, onlookers assess their dances of employed bees to choose potential sources, and scouts explore new locations. This algorithm balances exploration and exploitation through distinct roles. In PCG, ABC has been shown to be effective in tasks such as race track [151] for video games. By leveraging the principles of collective intelligence observed in natural bee colonies, ABC

contributes to creating diverse and engaging gaming environments.

B.3 Method

Procedural content generation (PCG), which includes characters, levels, and storylines, has become increasingly important in video games. Traditionally, genetic algorithms (GAs) have been the main optimization methods in this field. However, recent studies have indicated that other metaheuristic search algorithms may provide notable benefits in terms of their effectiveness and speed. This study attempted to illuminate this potential by thoroughly assessing and analyzing the efficacy of different metaheuristics in PCG challenges.

In this study, a computational approach was employed to address the research objectives. The primary methodology involves the application of GA, PSO, and ABC to a race track game and map generation. The process involves 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 and map layout. The study employed an experimental design with a primary focus on determining significant differences between the algorithms. The experimental setup ensured that all variables were held constant, with the only variable being the different algorithms used. This approach facilitates 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.

In our initial experiments, we used conventional populations. However, these experiments did not yield significant performance improvement. Therefore, we investigated the effects of a larger population of 600 individuals over 200 iterations. This choice, although seemingly unconventional, was motivated by the inherent complexity of the problem. The problem domain involves discovering diverse race tracks and maps that feature various curves to enhance player engagement. We hypothesized

that a larger population would provide broader exploration space, potentially leading to improved performance. However, it is important to acknowledge the trade-off between population size and computational cost. Larger populations require greater computational resources.

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

A systematic approach is employed in this study. We applied the chosen algorithms (GA, PSO, and ABC) to well-defined problems to generate racetracks and maps. Relevant analyses were performed to evaluate their performances. The experimental design emphasized consistency by controlling the variables across algorithms (e.g., population size and iteration count) to isolate the impact of each algorithm. This focus on algorithmic differences allowed us to gain insights into their comparative effectiveness in addressing the research question.

The detailed methodologies and specific results of these experiments are presented in our research papers [151] and [170]. The following section discusses the general trends observed in the experiments.

B.4 Data Acquisition

In Studies #1 [151] and #2 [170], we examined the effectiveness of three widely used metaheuristic algorithms in the realm of PCG, namely GA, PSO, and ABC. In GA, three different selection approaches, Tournament, Biased, and a mix of Tournament and Biased, were utilized, whereas in PSO, two variations of the PSO algorithms were implemented with two different values of the $C2$ acceleration coefficient. For both implementations, the value assigned to $C1$ was 2. The first implementation, denoted as PSO#1, uses a $C2$ value of four, and the second implementation, denoted as PSO#2,

Table B.1: Algorithms' performance in race track generation Task

	Best	Mean	Worst	SD
PSO#1	106.88	175.5	243.64	38.92
ABC	113.98	125.9	138.67	8.73
PSO#2	115.86	166.4	194.60	23.17
GA(Mix)	121.75	233.09	299.47	52.35
GA(Tournament)	137.74	182.9	225.63	25.28
GA(Biased)	304.30	370.1	443	46.13

uses a $C2$ value of five.

The following subsections discuss the data obtained from the two prior studies. Study#1 focuses on applying these algorithms to the task of generating race tracks, whereas study#2 examines their performance in map layout generation. The results obtained in both studies offer valuable insights into the strengths and weaknesses of each algorithm when applied in various PCG applications.

B.4.1 Race Track Application (Study#1)

In Study#1, we analyzed PSO, ABC, and GA in their application to the creation of race tracks. Our aim was to assess the strengths and weaknesses of each algorithm to determine which algorithm performs best in terms of solution quality and convergence speed for this type of game development.

Table B.1 and Fig. B.1 provide a comprehensive overview of the fitness values obtained by the chosen algorithms, arranged in descending order. The lowest fitness value represents the optimal solution that is most favorable. According to the data, PSO#1 achieved the highest optimal fitness based on this criterion.

According to the data presented in Table B.1, PSO#1 demonstrates strong performance, attaining the lowest best fitness value at 106.88 and exhibiting moderate variability at 38.92. This particular iteration of the PSO algorithm shows its exceptional ability to generate race tracks by locating optimal solutions. Additionally, PSO#1's relatively low standard deviation of 38.92 underscores its consistent and stable performance throughout the iterations. The efficiency with which PSO#1

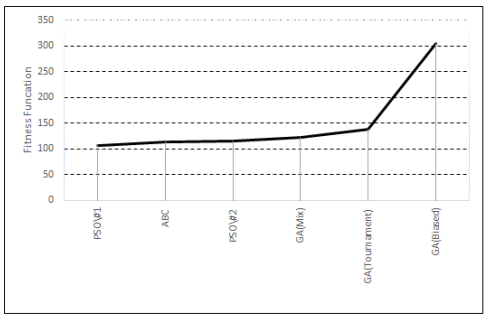


Figure B.1: Algorithms' performance in race track generation

explores the solution space and converges towards optimal solutions makes it a promising option for addressing PCG tasks.

Furthermore, ABC achieved a commendable best fitness value of 113.98, demonstrating its effectiveness in determining near-optimal solutions. The low standard deviation of 8.73 suggests that ABC maintains a high level of consistency in its performance, with minimal variability across iterations. This stability indicates that ABC is reliable and robust in its ability to search for optimal solutions, which makes it a valuable algorithm for PCG applications.

Although PSO#2 achieved a slightly higher best fitness value of 115.86 compared with PSO#1 and ABC, its performance remained competitive. However, the moderate standard deviation of 23.17 suggests some variability in fitness values across iterations, indicating slightly less stable performance compared to PSO#1 and ABC. Nonetheless, the ability of PSO#2 to effectively explore the solution space and converge towards near-optimal solutions makes it a viable choice for PCG tasks.

However, the GA with the ``Mix" selection approach achieved the best fitness value of 121.75, of GA variations indicating its ability to find relatively good solutions for the given task. However, the relatively high standard deviation of 52.35 suggests considerable variability in fitness values across iterations, indicating less stable performance compared to PSO#1, ABC, and PSO#2. Despite this variability, GA (mix) still demonstrates the potential for generating diverse and engaging content in PCG applications.

Move to GA (Tournament), This variant of the GA achieved a best fitness value of 137.74, which is higher than those of all other selected algorithms in this study.

Although the standard deviation of 25.28 suggests moderate variability in fitness values, a higher best fitness value indicates inferior performance compared to PSO#1, ABC, and PSO#2. The GA (Tournament) may still offer value in certain contexts, but may require further optimization for optimal performance in PCG tasks.

The last version of GA is GA (Biased) which showed the highest best fitness value of 304.30 among all algorithms in this research, the GA with the ``Biased'' selection approach demonstrates the poorest performance in finding optimal solutions. The moderate standard deviation of 46.13 suggests some variability in fitness values, indicating a relatively stable but suboptimal performance compared with the other algorithms. Although the GA (biased) may have limitations in its current form, further experimentation and optimization efforts could potentially improve its performance in PCG tasks.

The analysis of the results revealed valuable insights into the performance of various metaheuristic algorithms in PCG. PSO#1 and ABC have emerged as top performers, demonstrating superior performance in finding optimal solutions with high stability and consistency. Meanwhile, GAs show potential, but may require further refinement for optimal performance. (PSO) excels in finding optimal solutions by simulating the behavior of a swarm of particles moving through a solution space. This algorithm is particularly effective for PCG tasks because of its ability to efficiently explore the solution space and converge towards optimal solutions.

This finding contradicts established research [171] and lacks sufficient justification for the prevalent adoption of GAs in PCG across various domains, particularly in game-level generation. Search-based PCG, a specific subset within PCG, is commonly addressed using generate-and-test methodologies, such as GAs [172]. These studies collectively validated the utility and efficacy of GAs for PCG, emphasizing their capability to produce diverse and complex contents.

B.4.2 Map Generation Application (Study#2)

In Study# 2, we performed a thorough examination of the GA by employing various selection techniques, including hybrid, biased, and tournament, as well as PSO, with two distinct cognitive parameters (C2). In addition, we investigated the ABC method.

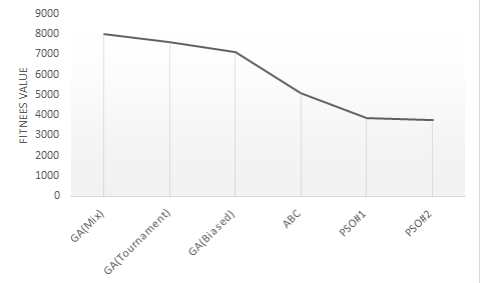


Figure B.2: The algorithms' performance in map generation

Table B.2: Algorithms' performance in map generation task

	Best	Mean	Worst	SD
GA(Mix)	7998	6128	5236	807.312
GA(Tournament)	7626	6207.6	5720	535.23
GA(Biased)	7110	5780	5040	631.17
ABC	5110	4519	4095	315.10
PSO#1	3835	2799.87	1128.4	1083.09
PSO#2	3780	2716.93	1156.9	997.53

This subsection presents essential information regarding the effectiveness and suitability of these algorithms in generating PCG maps through precise experimentation and assessment.

Study#1 demonstrated that PSO is effective for generating race tracks [151]. However, a distinct pattern emerges in the map layout generation results. As depicted in Fig. B.2, While PSO and ABC algorithms surpassed other methods in race track generation, their performance in terms of map quality fell short of expectations. This observation highlights the varying performance of different methods in different PCG tasks and underscores the adaptable nature of these algorithms.

Table B.2 lists the essential performance metrics, including the best fitness value, mean, worst fitness value, and standard deviation (SD). Notably, the highest fitness value signifies a high-quality map, providing valuable insight into the algorithm's ability to generate optimal solutions for the selected PCG task. An important finding from the data in Table B.2 is that GA(Mix) consistently achieved the highest fitness value in the best-case scenario, demonstrating its efficiency in generating solutions close to the optimal outcome. With an impressive best fitness value of 7998,

GA(mix) has emerged as a compelling candidate for delivering high-quality PCG solutions. Furthermore, the algorithm's comparatively high standard deviation of 807.312 suggests that the GA(Mix)'s performance might be less consistent across repetitions than the other algorithms. Further investigation is required to determine the impact of this inconsistency on the quality of the generated maps and their suitability for PCG tasks.

GA(tournament) and GA(biased) exhibited excellent performance characteristics. They achieved high fitness values of 7626 and 7110, respectively. Additionally, their standard deviations of 535.23 and 631.17 indicate relatively stable performance, suggesting consistency in generating results. This stable performance with good fitness values suggests a potential balance between exploration (finding new possibilities) and exploitation (focusing on promising areas) within the tournament (GA) and biased (GA) optimization processes. While neither achieved the highest absolute fitness value, these algorithms might be well suited for scenarios where consistent and reliable map generation is crucial.

In contrast, PSO#1 and PSO#2 exhibited lower performances than the other algorithms. Their best fitness values were 3835 and 3780, respectively, and their standard deviations were considerably higher (1083.09 and 997.53). These results indicated that both PSO variants displayed inconsistencies across ten repetitions. Although PSO has demonstrated effectiveness in other PCG tasks, such as generating race tracks, its performance in map generation appears to be less efficient and reliable than the GA variants explored in this study. Further investigation is required to understand why PSO struggles in this context.

Our findings regarding the performance of PSO in map generation appear to contradict the previous research on Evolutionary Game-based PSO (EGPSO) by Liu et al. (2008) [173]. EGPSO effectively addresses premature convergence and achieves robust convergence by incorporating an evolutionary algorithm into the PSO framework. However, in this study, we utilized a standalone PSO variant for a direct comparison with the GA and ABC algorithms. While ABC's performance resulted in maps of average quality (fitness value of 5110), it exhibited good stability in terms of standard deviation. The standard deviation, at its lowest value of 315.10, indicates consistent results when generating map layouts. This suggests that ABC may be a reliable

choice for applications where consistent map generation is important.

The performance and features of GA, PSO, and ABC used in map layout generation were thoroughly examined in Study #2 [170]. This analysis helps researchers and game developers to choose the most suitable algorithm for their specific needs and goals in different PCG tasks. These results serve as a foundation for further research. Combining these algorithms, each with its unique strengths, holds promise for the future. By carefully modifying variables, such as crossover frequency, mutation rate, and population size, we can create more effective and adaptable content creation techniques.

B.5 Implications for Different Applications

According to Shaker(2016) [1] and Sturtevant (2018) [3], the procedural content generation (PCG) field is defined by a wide range of approaches and strategies. The main focus of these studies was to highlight the diversity of the available methodologies. This is a subject that is also present in our study, highlighting the lack of a single algorithm suitable for all PCG problems. This alignment is consistent with the observed variations in algorithms, such as the GAs performance. Although GAs are effective in many situations, they have drawbacks, particularly when it comes to tasks such as race tracks. Hence, other approaches such as ABC and PSO should be investigated. Accepting this diversity of methods can lead to the creation of more complex and efficient solutions adapted to the specific needs of each PCG situation.

The features of a given task significantly affect the effectiveness of the algorithm. For instance, PSO excels at the exploration stage of building a race track, whereas ABC focuses on a precise track design. GA is far more adept at navigating complicated map layouts than PSO when it comes to map creation. These findings demonstrate the importance of understanding the particulars of quality standards, complexity of cases, and other distinctive features.

When selecting an algorithm, tradeoffs are always involved. PSO's rapid exploration of PSO may yield results more quickly; however, the quality of the solutions may suffer. However, development may progress more slowly as a result of ABC's

meticulous methodology. A thorough understanding of these trade-offs facilitates an informed decision-making process customized to the specific priorities of the PCG task.

As such, a ``Task-Specific Considerations'' approach promotes adopting a context-aware selection framework in place of a ``one-size-fits-all'' approach. The best algorithmic decisions may be made to produce the desired results by carefully examining the specifics of each PCG, identifying the advantages and disadvantages of the available algorithms, and assessing related trade-offs.

Beyond the algorithms discussed, further research efforts could reveal new metaheuristic or hybrid approaches, thereby expanding the range of possibilities for various PCG tasks.

Moreover, algorithmic performance is significantly affected by the manner in which the problem is described and the definition of the evaluation metrics. It is crucial to ensure that these elements represent the intended goals of the PCG task precisely.

Finally, utilizing domain knowledge relevant to a particular PCG domain, such as the genre of games or content type, can provide invaluable insights into relevant task attributes and possible algorithmic fit.

B.6 Conclusion

Although search-based PCG with metaheuristic algorithms such as GA, PSO, and ABC demonstrates promise, a crucial gap remains in understanding their interaction with specific PCG tasks. This study addresses this gap by analyzing the performance of these algorithms in race track and map generation tasks, based on our prior investigations. Our study concluded that while various algorithms can be explored for PCG, selecting the best approach requires careful consideration of the task-specific

factors.

The optimal choice depends on the unique characteristics of the PCG task, desired outcomes, and the potential trade-offs involved. Our findings offer valuable insights into the performance of context-dependent algorithms when applied to different optimization tasks.

Although PSO demonstrated exceptional robustness when applied to generate race tracks, its initial exploration phase frequently produced high-quality tracks, suggesting potential room for further optimization. This aligns with the observed performance of the GA(tournament), which effectively balances exploration and exploitation. However, when time constraints were not a critical factor, the thorough and consistent exploration of ABC proved advantageous for comprehensively exploring the search space for potential track configurations.

When the task was shifted to generating maps, PSO displayed poor performance in handling the complexities of the map layouts. However, ABC has emerged as a moderate choice, achieving a faster convergence and generating high-quality map content. These findings highlight the potential of ABC as an effective tool for developing complex maps.

These opposing findings raise the question of the assumption that GA dominates PCG universally [50] and motivated us to adopt a context-aware approach to algorithm selection rather than a single approach. Each algorithm has its advantages and disadvantages. When faced with time constraints, PSO's quick first investigation proves helpful; however, ABC's cautious approach excels when faced with strict quality criteria. Realizing these trade-offs and appropriately adjusting the selection allows PCG to reach its maximum potential.

This study highlights the significance of a careful study of each PCG task considering the complexity, quality requirements, and desired solution speed. It recommends a context-aware framework. Moreover, it promotes research beyond con-

ventional GAs, and highlights the distinct advantages of ABC and PSO. Concerning exploration, exploitation, consistency, and solution quality, each algorithm has unique benefits and drawbacks, and wise decisions should be made in accordance with project-specific requirements.

Context is essential. Every PCG work requires a unique analysis that uses domain expertise, such as game genre and content type, to improve the algorithmic fit. Most importantly, particular algorithmic strengths and weaknesses depend on the properties of the tasks themselves; therefore, future research should be guided by limitations within this study, such as the specific tasks investigated.

Through the integration of these perspectives, researchers and developers will be able to make knowledgeable choices about algorithms that optimally address their distinct PCG requirements, finally producing engaging material that is customized to their particular goals.

Appendix C

Code Source

This appendix provides access to the source code used in this thesis for various Procedural Content Generation (PCG) experiments. Each link below corresponds to a specific algorithm and task combination:

1. **Race Track Generation:**
    - **Genetic Algorithm (GA): https://github.com/AlyaseriSana/PCG-Track-GA-Method**
    - **Particle Swarm Optimization (PSO): https://github.com/AlyaseriSana/PCG-Track-PSO-Method**
    - **Artificial Bee Colony (ABC): https://github.com/AlyaseriSana/PCG-Track-ABC-Method**
2. **Map Layout Generation:**
    - **Genetic Algorithm (GA): https://github.com/AlyaseriSana/PCG-MapLayout-GA-Method:**
    - **Particle Swarm Optimization (PSO): https://github.com/AlyaseriSana/PCG-MapLayout-PSO-Method**
    - **Artificial Bee Colony (ABC): https://github.com/AlyaseriSana/PCG-MapLayout-ABC-Method**

Appendix D

DATA Results Source

This appendix provides access to the source data underlying the results presented in this thesis. The data generated in the various experiments are organized within the following GitHub repository:

- **Track generation experiments data: https://github.com/AlyaseriSana/PCG-DATA-ForTracks**

- **Map generation experiments data: https://github.com/AlyaseriSana/Metaheuristic-PCG**

Repository Contents:

The repository contains excel sheet for each experiment conducted, named according to the specific task investigated. The files containing the raw data generated by the experiments. These files in Excel format and has all ten runs output and the charts.

Appendix E

Symposium Presentation - Evaluating Alternative Metaheuristic Algorithms for Procedural Content Generation in Game Design

Presented at The Postgraduate Research Symposium, AUT, Septmber 2023

https://www.aut.ac.nz/research/postgraduate-student-support/postgraduate-research-symposium/presenters-at-the-postgraduate-research-symposium

Abstract

Procedural Content Generation (PCG) has emerged as a powerful approach for automating game content creation, offering significant benefits in terms of cost reduction and time efficiency compared to traditional game design and development processes [5]. While Genetic Algorithms (GAs) have been widely used in PCG, alternative metaheuristic algorithms such as Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) have demonstrated their effectiveness in delivering high-quality solutions and efficient optimization capabilities across different problem domains [15]. However, their application in PCG remains limited. I aim to evaluate the performance of PSO and ABC in map layout generation, challenging the conventional use of GAs. By comparing three metaheuristic algorithms (GA, ABC, and PSO) I seek to assess the effectiveness of these approaches in generating game levels and identify any obvious differences in their performance characteristics. Comprehensive experiments are conducted, applying GA, ABC, and PSO to a map layout generation. Metrics like convergence speed and content quality are used to evaluate the generated game content.

My findings reveal that both ABC and PSO demonstrate advantages over traditional GA implementations when generating game levels, indicating their potential for enhancing PCG. In this presentation, I will share the results of comparing three metaheuristic algorithms (GA, PSO, and ABC) in map layout generation for game levels, emphasizing the potential benefits of leveraging diverse algorithmic approaches to create more captivating and immersive game worlds. Also, I will conclude with a call for further research in this area to expose new possibilities in content generation. By considering varied metaheuristic approaches, game developers can improve content generation techniques and create more captivating and interactive player experiences.