Open Access
Baghdad Science Journal Vol.16 (Special Issue) 2019

DOI: http://dx.doi.org/10.21123/bsj.2019.16.2(SI).0515

An Analysis on the Applicability of Meta-Heuristic Searching Techniques for Automated Test Data Generation in Automatic Programming Assessment

Ja'afaru Musa [*] *Rohaida Romli* [**] *Nooraini Yusoff*

Received 30/9/2018, Accepted 14/11/2018, Published 20/6/2019

Abstract:

Automatic Programming Assessment (APA) has been gaining lots of attention among researchers mainly to support automated grading and marking of students' programming assignments or exercises systematically. APA is commonly identified as a method that can enhance accuracy, efficiency and consistency as well as providing instant feedback on students' programming solutions. In achieving APA, test data generation process is very important so as to perform a dynamic testing on students' assignment. In software testing field, many researches that focus on test data generation have demonstrated the successful of adoption of Meta-Heuristic Search Techniques (MHST) so as to enhance the procedure of deriving adequate test data for efficient testing. Nonetheless, thus far the researches on APA have not yet usefully exploited the techniques accordingly to include a better quality program testing coverage. Therefore, this study has conducted a comparative evaluation to identify any applicable MHST to support efficient Automated Test Data Generation (ATDG) in executing a dynamic-functional testing in APA. Several recent MHST are included in the comparative evaluation combining both the local and global search algorithms ranging from the year of 2000 until 2018. Result of this study suggests that the hybridization of Cuckoo Search with Tabu Search and lévy flight as one of promising MHST to be applied, as it's outperforms other MHST with regards to number of iterations and range of inputs.

Key words: Automatic Programming Assessment, Automated Test Data Generation, Metaheuristic Search Techniques.

Introduction:

Courses related to programming are viewed as the core subject of most of the Information Technology or Computer Science field (1). Therefore, in achieving the learning outcomes in this field, to be skillful in programming is vital. Strong foundation in principles of programming as well as significant practice in practical exercises are both necessary to be an expert in coding (2). Manual assessment on students' source codes and grading them in a large class is practically troublesome and time consuming to lecturers (3).

This is due to the facts that these activities lead to an extensive workload and burdensome tasks and most regularly foster unintended biases and distinctive standard of grading schemes. Hence, researches on Automatic Programming Assessment (or APA) that mainly aim at to support automated assessment and evaluation of students' programming exercises have gained much attention among scholars in this area. Practically, APA yields instant feedbacks and enhance the consistency, accuracy and efficiency of the assessment (4).

Therefore, many Automatic Programming Assessment Systems (or APAS) were produced such as Assyst (4), BOSS (5), TRAKLA (6), CourseMaster (7), SAC (8) and Bottlenose (9).

According to (3), activities of programming assessments are usually based on the similar concept used in software testing. Especially when executing a dynamic testing, it involves a process of running a program with various test cases or inputs to acquire the anticipated outputs (10). Therefore, a test data generation process assumes as a critical part in supplementing programming exercises assessments and also in APA. Test data generation is one of the procedures of deriving and generating test data which fulfills a given testing criterion (11).

School of Computing, Universiti Utara Malaysia, Malaysia
Corresponding authors:
[*]jaafaru_m@yahoo.com
[**]aida@uum.edu.my

In Software Testing (ST), the process of deriving test data is regularly a troublesome procedure and can be to a great degree requesting and difficult process (12). Hence, Automated Test Data Generation (ATDG) is viewed as a fascinating suggestion recommended among scholars in the previous year (13). As indicated by (14), the means of deriving various sets of test data for a program poses a couple of problems, such as the trouble in devising a full proof way to deal with guessing whether the outputs produced by the program are right; and conceivably in a few conditions that any two solutions will produce correctly a similar output. Thus, the selection of descriptive test data ought to be carefully derived to perform thorough and effective testing.

Recently, the adoption of any Meta-Heuristic Search Techniques (or MHST) in software testing researches has been demonstrated as one of the alternatives to effectively derive and generate an adequate and optimal test data for testing propose, and particularly to resolve issues dealing with combinatorial problems. Metaheuristics are commonly used to resolve optimization problems through the method of examining optimum solutions to a specific problem of importance (15). Moreover, the utilization of MHST would reason a better and promising solution because of the most optimum set of test data can guarantee the testing procedure can be executed effectively. This is due to the fact that one of the fundamental issues and difficulties in ATDG is the introduction to the non-deterministic polynomial hard problem or NP-hard (the time complexity of $O(n^n)$). (16) revealed the insights and patterns of the studies in ATDG between the year of 1976 and 2010. The outcome demonstrates that MHST have turned into the mainstream approaches applied since early year of 2000 and equally there have been expanding requests on defining the most optimal test case and test data in order to execute software testing productively and give enhancement on the quality of the final software product.

Integrating ATDG as a portion of testing in the field of Software Engineering (SE) or Computer Science (CS) education, subsequently there ought to be a promising prospective in acknowledging MHST in order to enhance the present studies of APA. So far, there have been a few researches that endeavored to incorporate ATDG and APA as revealed by (17). Nonetheless, in term of utilizing MHST, a study directed by (18) has had all the earmarks of being most of the related research. In regard to this limitation, this study aims to identify and afterward suggest any applicable MHST to efficiently realize ATDG in APA as an attempt to bridge the research gaps that occurred between CS education and ST areas. As existing studies in APA mainly focus on functional testing (16), hence our study focus is narrowed down to this type of testing.

The contents of this article are structured as follow: Section 2 briefly addresses some related works in regard to APA, ATDG, integrating APA and ATDG, ATDG and MHST, and MHST as a part of APA and ATDG; Section 3 provides a detailed discussion on the methodology used in this study; Section 4 presents and discusses the analysis and results of a comparative evaluation conducted for this study and finally, Section 5 concludes and suggests future research directions.

Related Works:

Although the problem of test data generation is a non-deterministic polynomial hard problem, scholars have tried to solve it using different methods. This section focuses on the background and previous studies that discussed about APA, ATDG, MHST, integration of ATDG and MHST, integration of APA and ATDG, and lastly utilizing MHST in an integrated APA and ATDG are summarized in the following sub-sections.

Automatic Programming Assessment (or APA)

Studies that are fundamentally identified with APA have been important to programming instructors since 1960s. Diverse approaches were utilized to evaluate different angles in students' program quality either based on static analysis or dynamic testing (white-box testing or black-box testing). In doing as such, a few program qualities are utilized, for example, the style of a program, reliability, correctness, complexity and many others. (16) outlined the sequence of researches done around this field of study by considering the said approaches. Its finding revealed that different strategies of assessment can be employed which may rely upon the particular goals or objectives of the course that are being taught, or the lecturers' own particular styles and preferences (19) or relying upon the schema of a program (20).

In a review paper written by (21), it included a few automated approaches for evaluating programming assignments that are also classified as static analysis and dynamic testing. Similarly, in another survey paper produced by (22), it emphasizes that APA systems were established on three generations that are the earliest system, tool-oriented system and web-oriented system.

Automated Test Data Generation (or ATDG)

Researches on ATDG are commonly focused on structural or functional-based test data generation approaches. (16) also presented the related studies done in this area in chronological order from the year of 1976 until 2010. The results show that the structural based test data generation that utilized path coverage criteria have gained the interests of most researchers as compared to the others. Also, it is indicated that since the early year of 2000 MHST have appeared to be the popular techniques applied in software testing as there has been many demands on searching the most optimal test case so as to achieve efficient software testing with certain extend of cost reduction. According to (23), in a review paper where he clarifies the approach of ATDG into two categories which are, test data generator and path selector. Furthermore, (24) classified the approach into three classes that are path oriented, goal oriented test data derivation and random. The approaches were classified based on either a dynamic or static testing and test case can be chosen from a distinctive level of software criteria.

Integrating APA and ATDG

There have been limited studies endeavored to incorporate both APA and ATDG. (6) introduced randomized input values to perform functional testing for APA. This technique is questionable since it cannot certify accuracy of the tested program and the scope of behaviours canvassed are frequently smaller in contrast with all the conceivable behaviours of the program. (25) recommended an outline to derive test case and test weightage to execute functional testing by utilizing Boundary Value Analysis (BVA) technique. The proposed schema has a limitation for an increasing number of input variables.

A study as proposed by (26) initiated a structural based test data generation method for APA. In this study, Java PathFinder (JPF) that is a software model checker was utilized to derive the desired test case. However, the method did not include negative testing criteria (27) to cover test data at error-prone points. On top of that, (28) also came accross with a similar method based on path coverage measures. Inclusion to that, the study also integrated functional-based test generation method to select and generate a satisfactory set of test case. Even the means of deriving the schema of test set is done automatically, the method of deriving of test data remains manually. Thus, it still requires a partial understanding in specific concepts of test case design. As established via a review paper by (29), it is indicated that the incorporation of MHST in the existing research of APA can perhaps provide further favorable and enhance outcome for future studies in this area.

ATDG and MHST

ATDG has dependably been a difficult issue in program testing research areas especially when dealing with the size and complexity (30). Nowadays, Artificial Intelligence (AI) techniques are changing the characteristic of the test automation (31, 32) and it becomes more efficient and cost effective (33). From the previous studies, it shows that in turn to effectively improve the excellence of the software test case generation, the comprehensive algorithm should be utilized. According to (12), MHST can be stretched out up to the intelligent investigation of program detail to help ATDG.

(34) presented in a review paper that MHST for ATDG are applied into two classification which are, population-solution and single-based techniques, the population-based have been recognized to entertain more preferred standpoint contrast to the Single-Solution based techniques. This review concluded that Genetic Algorithm (GA) is considered as the most generally used population-based MHST in ATDG. This is because the technique is among the well-known MHST. Also, population-based MHST has been recognized as a noteworthy procedure related in enhancing the present up-to-date of ATDG for program testing.

Utilizing MHST in an integration APA and ATDG

Table 1 presents the only study that integrates MHST and ATDG for a dynamic testing in APA. (35) stated that the presentation of MHST for ATDG as started since 1996. However, the combination of MHST and dynamic testing in APA has been initiated by (18). The study adapted a Particle Swarm Optimization (PSO) technique to automatically generate test cases to judge the correctness of students' C++ programming exercises particularly to execute a dynamic testing. Based on this finding, it may be established that the study of this area is still at its early stage. Therefore, the research towards this area needs to be investigate further as it furnish more optimal test data to support efficient testing in APA.

Table 1. Research that integrates MHST and ATDG in APA.

Author (s)	Testing Category	ATDG Approach	MHST	Program Quality factor
Foong et al. (18)	Dynamic testing	Fitness function	Particle Swarm Optimization	Correctness

Methodology:

In this research, a comparative evaluation was adapted in light of the principles suggested by Vartiainen (36), as an all-around process to identify, compare and report all the accessible studies and elite practices identified with a particular research question in such a way that is unprejudiced and repeatable. Figure 1 illustrates the processes involved in the comparative evaluation. Sub-sequence paragraphs detail out each process involved in the comparative evaluation.

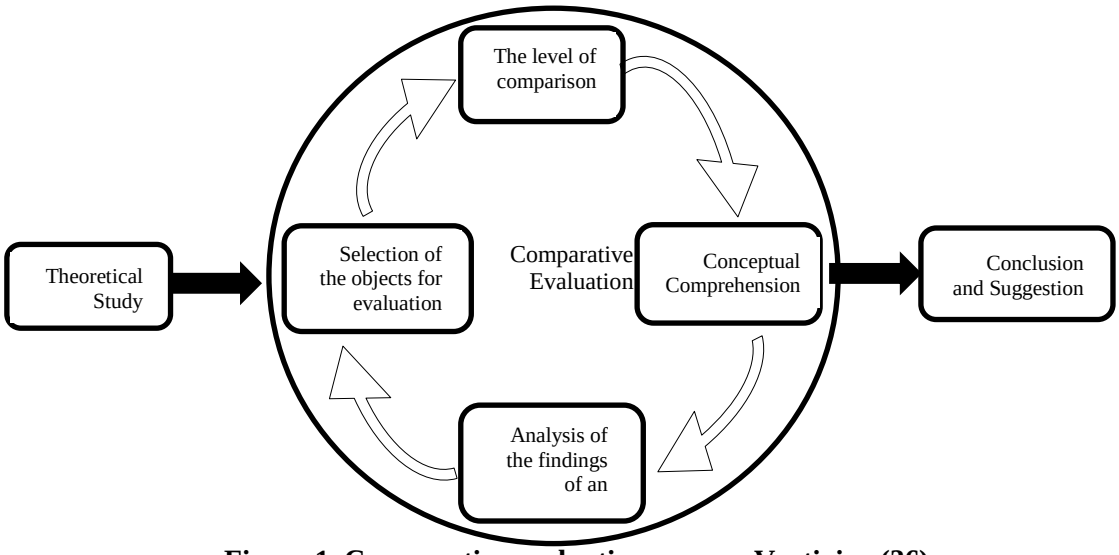

Figure 1. Comparative evaluation process Vartiaine (36).

Theoretical Study

A theoretical study consists of concepts, together with their definitions and existing theories that are used for particular study (37). Through theoretical study, it helps to understand the problems and finally it directs to formulate research objective and question. The method that was used in this phase is a literature survey that involves the review on existing works with respect to APA, ATDG, and an integration of ATDG and APA with the main focus on dynamic-functional testing. Also, particularly for researches with respect to ATDG, most of the focus is more on those studies that utilized MHST. The outcome of this theoretical study is the problem definition and research proposal.

Comparative Evaluation

Comparative evaluation is an instrument utilized to improve management and decision making (36). In this study, the comparative evaluation helps to analyze among the recent MHST and continue by suggesting any applicable MHST that has potential to support ATDG in APA. The method that was utilized in this phase is a comparative evaluation technique which consists of four activities:

a) Selection of the objects for evaluation – in this process, among of the recent MHST that are most commonly applied to support ATDG in software testing researches were identified. The identified MHST include Hill Climbing, Tabu Search, Simulated Annealing, Genetic Algorithm, Ant Colony Optimization, Migrating Birds Optimization, Particle Swarm Optimization, Artificial Bee Colony, Firefly Optimization, Harmony Search Algorithm, Cuckoo Search Algorithm, Bat Algorithm and Flower Pollination Algorithm.
b) Determination of the level of comparison- in this process, level of comparison to be used was identified as the components for comparison. These include year of the proposed study, the applied MHST, the proposed work, coverage technique applied, intermediate representation used (e.g. graph), evaluation parameters and benchmarks used and strength/weaknesses.
c) Conceptual comprehension - the concepts used in analyzing and comparing the selected

MHST were determined by number of iterations and range of inputs.

d) Analysis of the findings of an evaluation – through this process, findings uncovered in the comparative study activity made it possible to analysis and interpret the similarities and/or differences prevailing between the identified and evaluated MHSTs that are applicable to support ATDG in APA.

Conclusion and Suggestion

This phase is with regard to reporting the findings obtained from the conducted comparative evaluation in consequence to suggest any applicable MHST that has promising insight to support ATDG in APA. Hence, with the completion of this phase, the objective of this study has been achieved.

Analysis of Results and Discussion:

This section reveals the detailed analysis of results for the conducted comparitive evaluation and its discussion considering the related studies ranging from the year of 2000 until 2018. The first subsection explained in brief the basic concepts of each selected MHST. The tabularized comparison of all the studies are presented and discussed in the subsequence subsection.

Brief Concepts on the Selected MHSTs

In Computer Science and Mathematical optimization problem, MHST is an upper-level procedure to find or select a heuristic (discovery that employs a practical method not guaranteed to be optimal or perfect) to give a satisfactorily worthy outcome to an optimization problem, particularly with inadequate or flawed data or less computational capability (38). MHST for solving optimization problems can be categorized as Local Search and Global Search Algorithm. Figure 2 illustrates among of the selected MHST used in the conducted comparative evaluation by categorizing them based on local or global search algorithm.

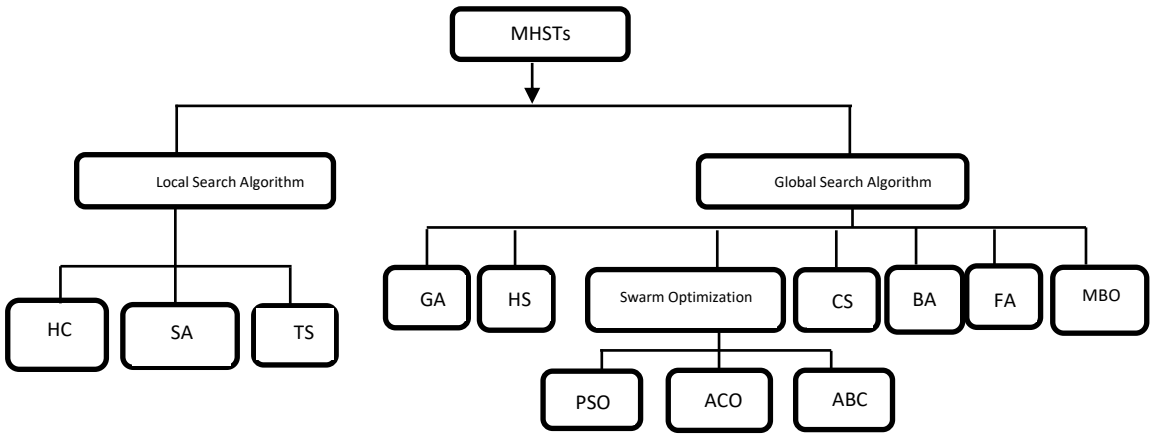

Figure 2. Selected Meta-Heuristic Search Techniques.

Local Search Algorithm: According to (39), local search algorithms try to search for a good solution among the neighboring solution and the local search terminates when the present solution is better than all its neighbors or could not get a better neighbor solution. These algorithms do not give attention to diversification. Table 2 listing out among of the selected MHST that fall under this local search algorithms with their brief concepts ad algorithms.

Table 2. MHST categorized as Local Search Algorithm.

Technique	Source and Year	Brief concepts	Algorithm
Hill Climbing (HC)	Ali et al. (40)	In this technique, the search start from a random selected solution by pondering the neighbors of the solution. Once a neighbor is discovered to be fitter than the others then it became the current solution in the search space and the procedure is recapped iteratively by changing a component of the solution until there is no fitter neighbor or current solution and the maximum probability	Step 1: Generate a possible solution Step 2: Compare newly generated Step 3: If the goal is achieved or no new states can be created, quit. Otherwise, return to step 1.
Tabu Search (TS)	Pirim et al. (41)	TS uses a neighbourhood search technique to repeatedly move from a solution *y* to a solution *y*' in the neighbourhood of *y*, till some terminating condition has been met. It changes the organization of each solution as the search proceeds and keep a search list or tabu list to prevent local optima problem.	Step 1: Generate initial solution y. Step 2: Initialize the Tabu List. Step 3: While set of candidate solutions Y‟ is not complete. Step 3.1: Generate candidate solution y‟ from current solution y Step 3.2: Add y‟ to Y‟ only if y‟ is not tabu or if at least one Aspiration Criterion is satisfied. Step 4: Select the best candidate solution y* in Y‟. Step 5: If fitness(y*) > fitness(y) then y = y*. Step 6: Update Tabu List and Aspiration Criteria Step 7: If termination condition met finish, otherwise go to Step 3.
Simulated Annealing (SA)	Pirim et al. (41)	SA technique randomly produces an enormous number of conceivable solutions, keeping both great and terrible solutions. As the procedure proceed with, the necessities for supplanting a current solution or remaining in the group becomes severe, imitating the moderate cooling of metallic annealing	Step 1: Compute randomly next position. Step 2: Determine the difference between the next position and current position, call this different delta. Step 3: If delta < 0, the assign the next position to the current position. Step 4: If delta > 0, then compute the probability of accepting the random next position. Step 5: If the probability is < the e^ (-delta / temperature), then assign the next position to the current position. Step 6: Decrease temperature by a factor of alpha. Step7: Loop to step 1 until temperature is not greater than epsilon

Global Search Algorithm: Global search algorithm starts to explore the search space by combining the strategies of intensification (exploitation) and diversification (exploration) to find global optima solution and the process stops when a user pre-defined stopping criterion is met (39). Table 3 listing out among of the selected MHST that are categoried as global search algorithm (with their brief concepts ad algorithms).

Table 3. MHST categorized as Global Search Algorithm.

Technique	Source and Year	Brief concepts	Algorithm
Genetic Algorithm (GA)	Bajeh and Abolarinwa (42)	The essential thought of GA is to first produce a preliminary population irregularly which comprise a unique answer for the issue called chromosomes, and afterward develop this population after various emphases called generations. Amid each generation, each chromosome is assessed, utilizing some measure of fitness. The creation of new offspring is done by either blending two chromosomes utilizing a crossover operator.	Step 1: Generate initial population. Step 2: Evaluate populations. Step 3: Apply Crossover to create offspring. Step 4: Apply Mutation to offspring. Step 5: Select parents and offspring to form the new population for the next generation. Step 6: If termination condition is met finish, otherwise go to Step 2.
Particle Swarm Optimization (PSO)	Eberhart and Kennedy (43), Weise (44)	PSO is a computational searching method used to optimize problems in an n-dimensional search space G. It is derived and motivated by the social behavior of birds, fishes and insects. These animals can produce a collective and efficient intelligence by exchanging very basic information about the environment in which they are exploring.	Step 1: Generate the initial population of particles Step 2: Initialize the position and velocity for the population with in the feasible region Step 3: Evaluate the fitness value of each particle Step 4: Update individual and global best fitness and positions Step 5: Update velocity and position of each particle Step 6: If termination condition is met finish, otherwise go to Step 3.
Ant Colony Optimization (ACO)	Singh et al. (45)	ACO is a probabilistic based computation problem algorithm which generates solutions by navigating graphs to find optimal solution. This Algorithm was inspired by the behavior shown by ants which are frantically searching for food. An ant lays a specific measure of pheromone trail along the way navigated while chasing for a food source.	Step 1: Initialization Step 2: Construction Step 3: Update Pheromones Step 4: If the goal is achieved, quit. Otherwise, return to step 2.
Harmony Search Algorithm (HSA)	Geem et al. (46)	HSA is inspired by musical concert activities that happen when a musician looks for an enhanced condition of harmony, for example, that amid jazz act of spontaneity. Jazz act of spontaneity searches to discover impeccable harmony (a faultless state) as controlled by an aesthetic standard, similarly as the optimization procedure tries to locate a global solution (an impeccable state) dictated by a target function.	Step 1: Initialize the HS Memory (HM). Step 2: Improvise a new solution from the HM. Step 3: Update the HM. The new solution from Step 2 is evaluated. Step 4: Repeat Step 2 to Step 3 until a pre-set termination criterion.
Firefly Algorithm (FA)	Yang (47)	This algorithm was presented by Xin at Cambridge University in 2007 which was propelled by the mating or blazing examples and conduct of fireflies. Fireflies are equipped for delivering biological light to pull in a partner or prey. There are about 2000 types of firefly which deliver brief and periodic flashes. These flashes regularly seem, by all account, to be in a one of a kind example and create an astonishing scene in the tropical territories amid summer. The intensity (I) of flashes diminishes as the distance (r) increments and along these lines most fireflies can convey just up to a few hundred meters.	Step 1: Initialize objective function of f(xi), where x= (x1, ….., xd)^T. Step 2: Generate the initial population of fireflies or xi (i=1, 2, …, n). Step 3: Determine the light intensity of Ii at xi via f(xi). Step 4: Calculate the attractiveness of fireflies. Step 5: Movement of less brighter fireflies towards brighter one. Step 6: Update the light Intensities, rank the fireflies and find the current best.

Table 3. MHST categorized as Global Search Algorithm (cont.).

Technique	Source and Year	Brief concepts	Algorithm
Artificial Bee Colony (ABC)	Karaboga (48)	ABC is a computational algorithm that is propelled by the foraging comportment of the bees when they are looking for their food source. The goals of bees are to found places of food source with highest nectar amount. The artificial honey bees are arranged into three gatherings: onlooker bees, employed bees and scouts. A bee sitting tight in the hive for making choice to pick a nourishment source is named as an onlooker. A bee that is presently scanning for food or exploiting a nourishment source is called an employed bee. Employed bees whose food sources can't be enhanced through a foreordained number of trials progress toward becoming scouts and their food sources are relinquished.	Step 1: Initially food source population is generated. Step 2 Evaluate the fitness function values. Step 3: The current best solution is evaluated. Step 4: In Employed Bee phase, produce a random solution. Step 5: Evaluate the fitness function of new solution through employed bee. Step 6: In Onlooker Bee phase, produce a random solution. Step 7: According to onlooker bee evaluates the fitness function of new solution. Step 8: Probability of occurrence of each food source is calculated. Step 9: Solutions having higher probability factor are improved. Step 10: Evaluate fitness functional value of the solution and memorize best solution. Step 11: Continue the process until termination condition reached.
Bat Algorithm (BA)	Yang (49)	BA is a metaheuristic computational algorithm that is inspired by the echolocation (is a type of sonar that guides bats in their flying and hunting behaviour) behavior of micro bats with emission of varying pulse rate and loudness. In BA, every bat is determined with a location x^t_i, at iteration t, and a velocity v^t_i in a d-dimensional search.	Step 1: Initialize and evaluate the initial population. Step 2: Generate the new solution by flying randomly. Step 3: Local search step. Step 4: Evaluate the new solution. Step 5: Save the best solution conditionally. Step 6: Find the best solution and update.
Cuckoo Search (CS)	Yang and Deb (50)	CS is propelled by the reproducing conduct of some cuckoo species that lay their eggs in the homes of host birds. Each egg signifies a solution, and a cuckoo egg signifies a new solution. The objective is to utilized the new and conceivably upgraded solutions (cuckoo eggs) to supplant worse solutions in the homes (host eggs).	Step 1: Initial a population of n host nests $\mathbf{x}_i$ (I = 1, 2,…,n). Step 2: Get a cuckoo (say i) randomly by Lévy flights. Step 3: Evaluate its fitness/quality F_i. Step 4: Choose a nest among n (say j) randomly. Step 5: Replace j by the new solution. Step 6: Keep the best solutions. Step 7: Rank the solutions and find the current best.
Flower pollination Algorithm (FPA)	Yang (51)	FPA was propelled by the fertilization procedure of flowers. Flower pollination is identified with the exchange of pollen, which is done by pollinators, for example, birds, insects, bats or wind. Some type of flowers has extraordinary pollinators for successful pollination. The target of the flower pollination process is the optimal reproduction of plants as far as numbers and also fittest.	Step 1: Initialize a population of n flower with random solutions. Step 2: Find the current best solution. Step 3: Check if termination condition is satisfied, otherwise go to Step2. Step 4: Output the best solution.
Migrating Birds Optimization (MBO)	Duman et al. (52)	MBO is a population-based neighborhood seek algorithm that is enlivened by the V arrangement flight of the moving birds which as turned out to be an efficient arrangement in energy sparing amid.	Step 1: Generate n initial solutions in V-formation. Step 2: improve the leading solution by generating and evaluating k neighbors of it. Step 3: Try to improve S_r by evaluating (k - x) neighbors of it and x unused best neighbors from the leader. Step 4: Move the leader solution to the end and forward one of the solutions following it to the leader position. Step 5: Update the best solution, otherwise return to Step 2.

In spite of the fact that metaheuristics are distinctive as some of them are global search based (GA, PSO, ACO, ABC, etc.), and others are local search based (HC, TS, SA), and despite the fact that they depend on various methods of insight, the mechanisms to effectively explore a search space are altogether in light of exploitation and exploration. In any case, it is conceivable to recognize "subtasks" in the search process where some metaheuristics outperform others. Among the approaches, the hybrid method outperforms and avoid sucking in local optimal even for larger dimension problems.

Analysis of MHSTs Implementation in ATDG

The tabulated comparison of the studies that implemented MHSTs in ATDG is presented in this subsection considering the trend of studies from the year of 2000 until 2018 and the category of MHST either local or global search algorithms. According to (16), the implementation of MHST in ATDG has become a popular technique in the field of software testing research since the early year of 2000. Table 4 and Table 5 illustrate the mentioned comparison based on the selected MHST.

Table 4. Comparison of MHST based test data generation (local search algorithms).

Author(s) and year	Proposed Work	Coverage Technique	Intermediate representation used (e.g. graph)	Evaluation parameters (EP) and benchmarks (BM) used	Strength/ Weaknesses
		Simulated Annealing (SA)			
Liu et al. (53)	Proposed a modified genetic algorithm by including Delaunay triangulation network in software	NA	NA	EP: number of branch BM: GA and M-GA	Quicker convergence speed and higher TDG proficiency however not exactly perfect in time proficiency and the derivation of unique test data.
Varshney and Mehrotra. (54)	Proposed an approach based on GA to generate test data for a program	Data flow coverage	Data flow coverage	EP: input range BM: GA and random	100% coverage achieved
Garg et al. (55)	Proposed SGA and HGA with new fitness function (ExLB)	Branch coverage	Control flow graph	EP: Fitness function BM: HGA and SGA	HGA give improved outcome than simple genetic algorithm combining extended level branch fitness value however couldn't cover the goal path.
Khan et al. (56)	Proposed an method for du-path coverage with genetic algorithm	Du-path coverage	Data flow coverage	EP: number of input	100% coverage achieved
Bao et al. (57)	Proposed an ATDG technique based on an improved GA	Path coverage	NA	EP: number of input and code size BM: IAGA, IGA, GA and random	It provides a more effective way to perform path testing
		Tabu Search (TS)			
Diaz et al. (58)	Proposed a chaining approach using TS for TDG	Branch coverage	Control flow graph	NA	Effective method for getting high branch coverage
Diaz et al. (59)	Proposed a TSGen with chaining approach for TDG	Branch coverage	Control flow graph	EP: input number BM: TSGen and random	Effective method for obtaining very 100% branch coverage
Rathore et al. (60)	Proposed a hybridized approach for TDG	Branch coverage	Control flow graph	EP: input number BM: GA and HTS	This approach outperforms the simple GA or TS
Karambir. (61)	Proposed a hybridized approach for ATDG	Path coverage	Control flow graph	NA	GTSA provide an effective path testing

Table 5. Comparison of MHST based test data generation (global search algorithms).

Author(s) and year	Proposed Work	Coverage Technique	Intermediate representation used (e.g. graph)	Evaluation parameters (EP) and benchmarks (BM) used	Strength/ Weaknesses
		Genetic Algorithm (GA)			
Tracey (62)	Proposed Random, Simulated annealing, GA for test data generation	Branch coverage	NA	NA	It generates higher quality of test data
Hermadi and Ahmad (63)	Proposed GA and path traversal approach, neighborhood effect, weighting, and normalization	Path coverage: path-wise and predicate-wise CFG	Control flow graph	NA	This integration improves the search space and achieved minimum time.
Andreou et al. (64)	present a method for ATDG centered on data flow coverage benchmarks.	Data flow coverage	Data flow graph	EP: code size and time	100% coverage achieved
Cao et al. (65)	Proposed an approach for deriving test case for a specific single path.	Path coverage		EP:Fitness function and time	The approach generate a great number of valid test data.
Srivastava and Kim. (66)	Identify the most crucial path cluster in a software	Path coverage	Control flow graph	EP: Fitness function	Enhance cost and exertion estimation in the testing stage.
Rauf et al. (67)	Proposed a coverage analysis method for GUI	Path coverage analysis	Event flow graph	EP: Fitness function	An enhanced coverage with an increased in data generation was achieved
Zhang et al. (68)	Proposed a test case generation for path coverage with error discovery	Path coverage with fault detection	NA	NA	100% coverage achieved with faults detection
Mairhofer et al. (69)	Present a technique to generate test case for dynamic programming languages	Statement coverage-class and method	NA	EP: Input value and code size	Higher and faster statement coverage was achieved
Xin et al. (70)	Proposed another path issue and broadens Species per Path approach on dynamic test case derivation	Path coverage	NA	EP: Fitness and time	Higher and efficient quality of test data generation
Ferrer et al. (71)	Proposed an evolutionary algorithm to deal with prioritization of pairwise test cases	NA	NA	NA	It's reduce time and the cost of test case generation
Maragathavalli et al. (72)	Proposed an approach for test data generation for instrumented programs	Multiple paths coverage	NA	EP: Fitness and time	Gives efficient test data
Mahajan et al. (73)	An analysis of the effectiveness of applying GA for ATDG	Coverage	Control flow graph	EP: number of iteration and input BM: GA and random	It is efficient, but couldn't achieved 100% coverage
Singhal et al. (74)	An approach to apply GA to TDG which accept 5 integers inputs to get 1 output	NA	Control flow graph	EP: number of input	It produces a high quality of test data.
Liu et al. (75)	Proposed a modified genetic algorithm by including Delaunay triangulation network in software	NA	NA	EP: number of branch BM: GA and M-GA	Quicker convergence speed and higher TDG proficiency however not exactly perfect in time proficiency and the derivation of unique test data.
Varshney and Mehrotra (76)	Proposed an approach based on GA to generate test data for a program	Data flow coverage	Data flow coverage	EP: input range BM: GA and random	100% coverage achieved
Garg and Garg (77)	Proposed SGA and HGA with new fitness function (ExLB)	Branch coverage	Control flow graph	EP: Fitness function BM: HGA and SGA	HGA give improved outcome than simple genetic algorithm combining extended level branch fitness

					value however couldn't cover the goal path.
Khan et al. (78)	Proposed an method for du-path coverage with genetic algorithm	Du-path coverage	Data flow coverage	EP: number of input	100% coverage achieved
Bao et al. (79)	Proposed an ATDG technique based on an improved GA	Path coverage	NA	EP: number of input and code size BM: IAGA, IGA, GA and random	It provides a more effective way to perform path testing
			Particle Swarm Optimization (PSO)		
Windisch et al. (80)	Proposed an CL-PSO approach for TDG	Branch coverage	NA	EP: input parameter BM: CL-PSO and GA	CL-PSO outperforms GAs for most branch elements to be covered in terms of effectivenessand efficiency.
Zhu and Yang (81)	Proposed an Adaptive PSO approach for ATDG	code coverage	NA	EP: number of iteration BM: APSO, PSO and IGA	APSO outflanks GAs for most code components to be canvassed as far as adequacy and effectivenes.
Cui et al. (82)	Present an efficient ATDG method	NA	NA	EP: number of generation and input BM: GA and PSO	The integration of both method achieved efficient TDG
Singla et al. (83)	present a technique coordinating the strength of genetic algorithm and PSO called GPSCA	NA	Control flow graph	EP: number of iteration and input BM: GA, PSO and GPSCA	APSO is effective in term iteration times and coverage speed
de Souza et al. (84)	Proposed PSO in Multi-Objective test data derivation	NA	NA	NA	BMOPSO-CDR technique outflanked BMOPSO and the random strategy
Tiwari et al. (85)	Present a modified PSO-TVAC	NA	Control flow graph	EP: number of input and iteration BM: PSO-TVAC and modified PSO-TVAC	PSO-TVAC is effective in term iteration times and coverage speed
Huang et al. (86)	Proposed an improved PSO with self-activity feedback (SAF)	Path coverage	NA	EP: iteration number BM: GA, PSO, SAF-GPSO	PSO-SAF is more effective in multi-path derivation of test data compared to the standard PSO.
Jiang et al. (87)	Proposed Reduced Adaptive PSO for TDG	NA	Control flow graph	EP: input number BM: RAPSO, APSO and GPSCA	RAPSO is effective in term iteration times and coverage speed
Ting and Hui (88)	Present PSO based on clustering to generate test data	NA	Control flow graph	EP: iteration time and number of input BM: Cluster PSO, RAPSO and HC-MARPSO	Cluster PSO is more noteworthy than APSO, GPSCA, RAPSO and MARPSO in emphasis times, and has a more noteworthy benefit on the convergence speed.
Jatana et al. (89)	Proposed a PSO-MT method for TDG	NA	NA	NA	killed noteworthy amount of mutant
Kumar et al. (90)	Proposed Accelerating PSO to generate test case for data-flow dependencies	NA	Data flow	EP: number of iteration BM: APSO, GA, PSO and random	APSO gives efficient result compared to random search, GA and PSO in enhancing the convergence speed
			Ant Colony Optimization (ACO)		
Li et al. (91)	Proposed ACO method for generating UML test cases	State based	State chart diagram	NA	Non-duplicate and practical test cases are derived that attained all state coverage

					measures.
Farah and Harmanani (92)	Proposed a technique using ACO to select test pattern	Fault coverage	NA	NA	Set of high fault coverage test data were derived within minimum time
Chen et al. (93)	Proposed a test data approach	NA	NA	EP: run time	ACO is effective for TDG
Srivastava (94)	Proposed ACO to generate path	Path coverage	Control flow graph	NA	ACO achieved efficient and ideal path coverage
Bauersfeld and Wegener (95)	Present ACO for test sequence generation	NA	NA	NA	It's efficiently generate single sequence, but multi sequence generation needs more improvement
Bajaj et al. (96)	Proposed ACO approach for test data generation	Path coverage	Control flow graph	NA	Achieved 100% path coverage with little duplicate
Mao et al. (97)	Proposed a framework for TDG	Branch coverage	NA	EP: average time, generation and rate BM: GA, SA and ACO	TDG_ACO outperform SA and GA based on effectiveness and practicality
Alzubaidy and Alhafid (98)	Proposed ATDG	Path coverage	Control flow graph	NA	Achieved efficient coverage
Yang et al. (99)	Proposed improved ACO for TDG	Path coverage	NA	NA	Set of high fault coverage test data were derived within minimum time
Sharma (32)	Proposed improved ACO for TDG	NA	NA	NA	Achieved efficient coverage with less iteration
Arifiani and Rochimah. (100)	Proposed statistical testing	State based	State machine diagram	NA	Generate test data using ACO
		Artificial Bee Colony (ABC)			
Mala and Mohan. (101)	Proposed ABC for test suite generation	State coverage, code coverage, branch coverage and path coverage	NA	NA	Achieved near global optimal test suit with less iteration
Dahiya et al. (102)	Proposed test data generation using ABC	NA	Control flow graph	NA	It achieved excellent performance for most of the program expect enormous outcome areas and various equality-centered path limits
Lam et al. (103)	Present a novel search approach	NA	NA	NA	ABC is better compare to GA, ACO and Tabu search
Singh and Sandhu. (104)	Proposed ABC approach for test data generation	NA	NA	NA	ABC is effective for TDG
Suri et al. (105)	Proposed a hybrid approach based on GA and BCO	NA	NA	EP: number of iteration BM: NA	Ideal outcomes were attained in less time
Dalal and Chhillar. (106)	Proposed a hybrid approach based on GA and BCO	NA	NA	NA	Optimum results achieved in minimum time
Kumar et al. (107)	Proposed a test data generation solution using ABC approach	Path coverage	Control flow graph	EP: number of iteration BM: ABC, GA and PSO	ABC outperform GA and PSO and achieved good def-path coverage
		Cuckoo Search (CS)			
Srivastava et al. (108)	Proposed CS with lévy flight and TS for ATDG	Path coverage	Control dependency graph	EP: number of iteration and input BM: GA and TS, GA, CS and TS, and CS and TS with lévy flight	It performs (number of iteration, number of parameters and node coverage) is much better compared to the existing MHST such as GA, ABC, ACO etc.
Naseer et al. (109)	Proposed a pairwise approach for combinatorial test issue	NA	NA	NA	PairCS achieved an effective and efficiency performances

					than other computational-based strategies as well as nature-inspired algorithm such SA, PSO etc.
Ahmed (110)	Proposed a method to decrease the test suit in configuration-aware structural testing	NA	NA	NA	Perform better than other computational-based strategies and MHST
			Harmony Search (HS)		
Alsewari and Zamli (111)	Present a pairwise strategy for combinatorial testing	NA	NA	EP: iteration and test size BM: GA, SA, ACA, PPSTG and PHSS	PHSS outperform most strategies and future works can be done on the performance of PHSS
Xiang et al. (112)	Suggest a technique to less the number of test cases	NA	NA	EP: test size BM: Testcover, SigmaZone, Pro-test, jenny and HS-PTSGT	HS-PTSGT achieved optimum PTS with less test data
			Bat Algorithm (BA)		
Alsariera et al. (113)	Present a pairwise strategy for combinatorial testing	NA	NA	EP: test data size BM: TVG, PICT, CTE_XL, TConfig, IPOG, jenny, PPSTG, PHSS and BPTS	BPTS outperform most strategies and future works can be done on its effectiveness
Öztürk (114)	Proposed a test cases prioritization	NA	NA	EP: NA BM: PSO, greedy, ACO, LBS and BITCP	BITCP is excellent compared to other traditional approaches
			Flower Pollination Algorithm (FPA)		
Nasser et al. (115)	Proposed a t-way strategy for sequence and sequence-less test data generation	NA	NA	NA	The size of test suite produced is good.
Nasser et al. (116)	Proposed a technique for generating pairwise test suite	NA	NA	NA	In terms of generating pairwise test suite size PairFS is more efficient than other tradition strategies
			Firefly Algorithm (FA)		
Srivatsava et al. (117)	Proposed a technique to generate optimal test path	Path coverage	Control flow graph	NA	Generation of test path are critical and optimal
Sahoo et al. (118)	Proposed a method to generate and optimize random test case	NA	NA	EP: number of iteration and objective function. BM: PSO, HS, CS, BA and FA	Effective and efficient test data are produced
			Migrating Birds Optimization (MBO)		
Zakaria and Zamli. (119)	Proposed technique to generate optimal pairwise test data reduction	NA	NA	NA	iPMBOS performs better PMBOS when dealing with less test sized generation

In summary, while lot of studies have been conducted on ATDG using GA, however when compared to other algorithms like ACO, PSO, HS, ABC and so on with regards to number of iterations, GA records high number of iterations before reaching its optimum solution (120, 121). Furthermore, the study of Srivastava et al. (108), have shown better performance in reference to the number of iterations and inputs range when compared with other existing MHST, as its generate effective and optimally test data with less number of iterations. Further discussion on the produced comparison by categorizing based on the selected MHSTs is as follows:

i) Simulated Annealing (SA)

Not much research has been done on SA with regards to test case generation to the best of our knowledge. Researchers are encouraged to explore the research gaps in this technique. thus, more work can be done on other coverage criteria using SA such as Modified Condition /Decision Condition

Coverage (MC/DC), data flow testing, branch coverage and so on.

ii) Tabu Search (TS)

Tabu search might provide improved test data generation for regression testing, as it has memory function (tubelist). It is powerful in accomplishing 100% branch coverage. With regards to test data derivation based on other coverage criterion utilizing Tabu Search more studies are required. Tabu search might demerit in test case derivation for sizable and difficult issue as its absence of adequate and long-term memory. Therefore, more studies to propose improved memory strategy are needed.

iii) Genetic Algorithm (GA)

Based on the above investigations, GA and it extensions have been used as a popular MHST for ATDG. It is inferred that GA is an effective strategy for realizing software test data generation. It beats the thorough search and local search method. Be that as it may, in the meantime, it experiences substantial number of iterations and untimely convergence. In terms of basis path coverage, GA is not capable given promising outcome. According to Garg and Garg (55), they mentioned that the fitness function proposed in their research may produce proficient test cases with the integration of Adaptive PSO and CS as this techniques balance exploitation and exploration better or their algorithm might need better mutation rate, as their proposed algorithm have not covered a target path. Genetic algorithm is great at global search and simulated annealing is great at local search. Thus, the mix of genetic algorithm and simulated annealing might provide an improve path coverage.

iv) Particle Swarm Optimization (PSO)

PSO and extensions of PSO based TDG outperforms GA based TDG (80, 81, 83). A great deal of research has been done on structural testing utilizing PSO and its variation and it gives great outcome. Ting and Hui (88) established in their study that Cluster PSO effectively avoid getting trapped in local optima, however the application of fuzzy concept to each cluster may reduce the cost further. Some PSO variations like Emotional PSO, Fuzzy PSO have not been investigated with test data generation (122, 123).

v) Ant Colony Optimization (ACO)

ACO is generally utilized for systematic generation of test case and test pattern. It gives great outcome in test design or sequence generation. Likewise, deriving multiple input sequence with ACO is hard because of its pheromone updating technique. Because ACO testing is predominantly a sequential procedure and selection of the solutions are done just toward the end. ACO might offer better outcome when the algorithm is integrated with existing metaheuristic approaches.

vi) Artificial Bee Colony (ABC)

ABC technique has some benefits over GA, TS and ACO. Contradictory to ABC, ACO centered testing is a similar procedure and choosing of solution is done in every phase. Optimization overhead and memory restrain issues are very much adjusted in ABC compared with ACO. In ABC technique, local optimum issue does not happen. The primary drawback in TS is the condition of recalling entire test data for present search that makes a major issue, yet ABC does not need recollecting of entire test data. Expanding alteration rate is a significant issue in genetic algorithm technique. It produces temperamental outcome as it experiences untimely convergence. ABC consistently enhances the proficiency of test data as far as their path coverage degree. Pheromone updating is a generous overhead in ACO technique. In ABC, there is no overhead of updating pheromone. In spite of the fact that ABC has a few favorable circumstances over other metaheuristic algorithms, yet it's search space is constraint by initial solution. For more effective test data derivation ordinary dispersion sample might be utilized.

vii) Cuckoo Search (CS)

The search technique has demonstrated its preferred proficiency over the existing search technique like GA, PSO, ACO and ABC and so forth, thus, it turns into an attracting search strategy for scholars in software testing area. As it generates an efficient test suite (109). likewise, In the study of Srivastava et al. (108), the proposed algorithm proofed that as the range of inputs increases, the number of iterations is reduced as compared to other technique, hence CS generates an efficient and optimal test data. Furthermore, in order to reduce the time complexity for high collection integers (16,32 bits), the integration of a proficient data arrangements for Tabu lists and lévy flights can be utilized.

Conclusion and Suggestion:

In this paper, a comparative evaluation was conducted to identify the applicable MHST to support ATDG in executing a dynamic-functional testing in APA. This research does not include any unpublished study and the aim is on discovering the solutions to the identified research question. After carrying out the comparative evaluation that includes a sequence of stages, there were 64 selected studies recognized from the year of 2000

until 2018. According to the hybridized approached proposed by (108), an integration of Cuckoo Search with Tabu Search and lévy flight can be concluded to be a promising solution to support derivation of optimal test data for efficient ATDG in APA. To the best of our knowledge, there is no one single study that has accomplished an exact outcome for a program testing based on the searching techniques. More MHST can still be implemented to ATDG to accomplish a better outcome. Enough work on Harmony search (HS), Bat Algorithm (BA), Flower Pollination Algorithm (FPA), Firefly Algorithm (FA), Migrating Birds Optimization (MBO) have not been done yet on ATDG as a whole, thus more work are possible to accomplish more better outcome. However, much works have been done on structural testing and yet merely a couple of works focused on functional testing utilizing MHST. Test case generation is a multi-objective task however, a couple of work is done toward that path. Some more hybridization MHST are conceivable to conquer the limitation of other single MHST.

Acknowledgement:

The authors acknowledge Ministry of Higher Education FRGS Fund (Code SO: 13793 & 12821) of Universiti Utara Malaysia for supporting this work.

Conflicts of Interest: None.

References:

1. Curricula C. The Joint Task Force on Computing Curricula IEEE Computer Society and the Association for Computing Machinery.
2. Sommerville I, Sawyer P. Requirements engineering: a good practice guide. John Wiley & Sons, Inc.; 1997 Apr 1.
3. Jackson D. A software system for grading student computer programs. Computers & Education. 1996 Dec 1;27(3-4):171-80.
4. Jackson D, Usher M. Grading student programs using ASSYST. InACM SIGCSE Bulletin 1997 Mar 1; 29(1):335-339.
5. Luck M, Joy M. A secure on-line submission system. Software: Practice and Experience. 1999 Jul 10;29(8):721-40.
6. Malmi L, Karavirta V, Korhonen A, Nikander J, Seppälä O, Silvasti P. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. Informatics in education. 2004 Jul 1;3(2):267.
7. Higgins CA, Gray G, Symeonidis P, Tsintsifas A. Automated assessment and experiences of teaching programming. Journal on Educational Resources in Computing (JERIC). 2005 Sep 1;5(3):5.
8. Auffarth B, López-Sánchez M, Campos i Miralles J, Puig A. System for automated assistance in correction of programming exercises (sac). In International Congress University Teaching and Innovation (CIDUI) 2008 (pp. 104-113).
9. Sherman M, Bassil S, Lipman D, Tuck N, Martin F. Impact of auto-grading on an introductory computing course. Journal of Computing Sciences in Colleges. 2013 Jun 1;28(6):69-75.
10. Chu HD, Dobson JE, Liu IC. FAST: a framework for automating statistics-based testing. Software Quality Journal. 1997 Mar 1;6(1):13-36.
11. Korel B. Automated software test data generation. IEEE Transactions on software engineering. 1990 Aug;16(8):870-9.
12. Tahbildar H, Kalita B. Automated software test data generation: direction of research. International Journal of Computer Science and Engineering Survey. 2011 Feb;2(1):99-120.
13. Mao C, Yu X, Chen J, Chen J. Generating test data for structural testing based on ant colony optimization. InQuality Software (QSIC), 2012 12th International Conference on 2012 Aug 27 (pp. 98-101). IEEE.
14. Jackson D. A semi-automated approach to online assessment. ACM SIGCSE Bulletin. 2000 Sep 1;32(3):164-7.
15. Hussain K, Salleh MN, Cheng S, Shi Y. Metaheuristic research: a comprehensive survey. Artificial Intelligence Review. 2018:1-43.
16. Romli R, Sulaiman S, Zamli KZ. Automatic programming assessment and test data generation a review on its approaches. In Information Technology (ITSim), 2010 International Symposium in 2010 Jun 15 (Vol. 3, pp. 1186-1192). IEEE.
17. Romli R, Sulaiman S, Zamli KZ. Designing a test set for structural testing in automatic programming assessment. International Journal of Advanced Soft Computing and Application. 2013 Dec 1;5(3):1-24.
18. Foong OM, Tran QT, Yong SP, Rais HM. Swarm inspired test case generation for online C++ programming assessment. InComputer and Information Sciences (ICCOINS), 2014 International Conference on 2014 Jun 3 (pp. 1-5). IEEE.
19. Joy M, Griffiths N, Boyatt R. The boss online submission and assessment system. Journal on Educational Resources in Computing (JERIC). 2005 Sep 1;5(3):2.
20. Shukur Z. *The automatic assessment of Z specifications*(Doctoral dissertation, University of Nottingham).
21. Ala-Mutka KM. A survey of automated assessment approaches for programming assignments. Computer science education. 2005 Jun 1;15(2):83-102.
22. Douce C, Livingstone D, Orwell J. Automatic test-based assessment of programming: A review. Journal on Educational Resources in Computing (JERIC). 2005 Sep 1;5(3):4.
23. Edvardsson J. A survey on automatic test data generation. InProceedings of the 2nd Conference on

Computer Science and Engineering 1999 Oct 21 (pp. 21-28).
24. Ferguson R, Korel B. The chaining approach for software test data generation. ACM Transactions on Software Engineering and Methodology (TOSEM). 1996 Jan 1;5(1):63-86.
25. Shukur Z, Romli R, Hamdan AB. Skema Penjanaan Data dan Pemberat Ujian Berasaskan Kaedah Analisis Nilai Sempadan (A Schema of Generating Test Data and Test Weight Based on Boundary Value Analysis Technique). Technology Journal. 2005 Jun;42:23-40.
26. Ihantola P. Automatic test data generation for programming exercises with symbolic execution and Java PathFinder. Helsinki University of Technology. 2006 May 30.
27. IPL Information Processing Ltd. Structural Coverage Metrics. Available: http://www.ipl.com/pdf/p0823.pdf. 1997. Retrieved on: 10 Feb 2018
28. Romli R. Test data generation framework for Automatic Programming Assessment. PhD Thesis, Universiti Sains Malaysia, Malaysia, 2014.
29. Romli R, Mahzan N, Mahmod M, Omar M. Test Data Generation Approaches for Structural Testing and Automatic Programming Assessment: A Systematic Literature Review. Advanced Science Letters. 2017 May 1;23(5):3984-9.
30. Bertolino A. Software testing research: Achievements, challenges, dreams. In2007 Future of Software Engineering 2007 May 23 (pp. 85-103). IEEE Computer Society.
31. McMinn P. Search-based software test data generation: a survey. Software testing, Verification and reliability. 2004 Jun;14(2):105-56.
32. Sharma P. Automated Software Testing Using Metahurestic Technique Based on Improved Ant Algorithms for Software Testing. International Journal on Recent and Innovation Trends in Computing and Communication. 2014;2(11):3505-10.
33. Gautam A. An Overview of Automatic Test Data Generation and Meta-heuristic Search Techniques. International Journal. 2015 Apr;5(4).
34. Romli R, Nordin N, Omar M, Mahmod M. A Review on Meta-Heuristic Search Techniques for Automated Test Data Generation: Applicability Towards Improving Automatic Programming Assessment. InInternational Conference of Reliable Information and Communication Technology 2017 Apr 23 (pp. 896-906). Springer, Cham.
35. Jones BF, Sthamer HH, Eyres DE. Automatic structural testing using genetic algorithms. Software engineering journal. 1996 Sep;11(5):299-306.
36. Vartiainen P. On the principles of comparative evaluation. Evaluation. 2002 Jul;8(3):359-71.
37. Sekaran U. Research methods for business: A skill building approach. John Wiley & Sons; 2006.
38. Lakhotia K, Harman M, McMinn P. A multi-objective approach to search-based test data generation. InProceedings of the 9th annual conference on Genetic and evolutionary computation 2007 Jul 7 (pp. 1098-1105). ACM.
39. Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing. 2009 Jun 1;8(2):239-87.
40. Ali S, Briand LC, Hemmati H, Panesar-Walawege RK. A systematic review of the application and empirical investigation of search-based test case generation. IEEE Transactions on Software Engineering. 2010 Nov;36(6):742-62.
41. Pirim H, Bayraktar E, Eksioglu B. Tabu Search: a comparative study. InTabu Search 2008. InTech.
42. Bajeh AO, Abolarinwa KO. Optimization: a comparative study of genetic and tabu search algorithms. International Journal of Computer Applications (IJCA). 2011 Oct;31(5).
43. Eberhart RC, Kennedy J. Particle swarm optimization, proceeding of IEEE International Conference on Neural Network. Perth, Australia. 1995 Nov:1942-8.
44. Weise T. Global optimization algorithms-theory and application. Self-published. 2009 Jun 26;2.
45. Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization. ACM SIGSOFT Software Engineering Notes. 2010 Jul 20;35(4):1-7.
46. Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. simulation. 2001 Feb;76(2):60-8.
47. Yang XS. Firefly algorithms for multimodal optimization. InInternational symposium on stochastic algorithms 2009 Oct 26 (pp. 169-178). Springer, Berlin, Heidelberg.
48. Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department; 2005 Oct.
49. Yang XS. A new metaheuristic bat-inspired algorithm. InNature inspired cooperative strategies for optimization (NICSO 2010) 2010 (pp. 65-74). Springer, Berlin, Heidelberg.
50. Yang XS, Deb S. Engineering optimisation by cuckoo search. arXiv preprint arXiv:1005.2908. 2010 May 17.
51. Yang XS. Flower pollination algorithm for global optimization. InInternational conference on unconventional computing and natural computation 2012 Sep 3 (pp. 240-249). Springer, Berlin, Heidelberg.
52. Duman E, Uysal M, Alkaya AF. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences. 2012 Dec 25;217:65-77.
53. Liu D, Wang X, Wang J. Automatic Test Case Generation based on Genetic Algorithm. Journal of Theoretical & Applied Information Technology. 2013 Feb 10;48(1).
54. Varshney S, Mehrotra M. Automated software test data generation for data flow dependencies using genetic algorithm. International Journal. 2014 Feb;4(2).

55. Garg D, Garg P. Basis Path Testing Using SGA & HGA with ExLB Fitness Function. Procedia Computer Science. 2015 Jan 1;70:593-602.
56. Khan R, Amjad M, Srivastava AK. Optimization of automatic generated test cases for path testing using genetic algorithm. InComputational Intelligence & Communication Technology (CICT), 2016 Second International Conference on 2016 Feb 12 (pp. 32-36). IEEE.
57. Bao X, Xiong Z, Zhang N, Qian J, Wu B, Zhang W. Path-oriented test cases generation based adaptive genetic algorithm. PloS one. 2017 Nov 14;12(11):e0187471.
58. Díaz E, Tuya J, Blanco R. Automated software testing using a metaheuristic technique based on tabu search. InAutomated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on 2003 Oct 6 (pp. 310-313). IEEE.
59. Díaz E, Tuya J, Blanco R, Dolado JJ. A tabu search algorithm for structural software testing. Computers & Operations Research. 2008 Oct 1;35(10):3052-72.
60. Rathore A, Bohara A, Prashil RG, Prashanth TS, Srivastava PR. Application of genetic algorithm and tabu search in software testing. InProceedings of the Fourth Annual ACM Bangalore Conference 2011 Mar 25 (p. 23). ACM.
61. Karambir CS. Optimization of Basis Path Testing using Genetic Tabu Search Algorithm.
62. Tracey NJ. *A search-based automated test-data generation framework for safety-critical software* (Doctoral dissertation, University of York).
63. Hermadi I, Ahmed MA. Genetic algorithm based test data generator. InThe 2003 Congress on Evolutionary Computation, 2003. CEC'03. 2003 Dec 8 (Vol. 1, pp. 85-91). IEEE.
64. Andreou AS, Economides KA, Sofokleous AA. An automatic software test-data generation scheme based on data flow criteria and genetic algorithms. Incit 2007 Oct 16 (pp. 867-872). IEEE.
65. Cao Y, Hu C, Li L. An approach to generate software test data for a specific path automatically with genetic algorithm. InReliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on 2009 Jul 20 (pp. 888-892). IEEE.
66. Srivastava PR, Kim TH. Application of genetic algorithm in software testing. International Journal of software Engineering and its Applications. 2009 Oct;3(4):87-96.
67. Rauf A, Anwar S, Jaffer MA, Shahid AA. Automated GUI test coverage analysis using GA. InInformation Technology: New Generations (ITNG), 2010 Seventh International Conference on 2010 Apr 12 (pp. 1057-1062). IEEE.
68. Zhang Y, Gong D, Luo Y. Evolutionary generation of test data for path coverage with faults detection. InNatural Computation (ICNC), 2011 Seventh International Conference on 2011 Jul 26 (Vol. 4, pp. 2086-2090). IEEE.
69. Mairhofer S, Feldt R, Torkar R. Search-based software testing and test data generation for a dynamic programming language. InProceedings of the 13th annual conference on Genetic and evolutionary computation 2011 Jul 12 (pp. 1859-1866). ACM.
70. Xin Z, Hu L, Li N. The species per path approach to GEMGA-based test data generation. InMultimedia Technology (ICMT), 2011 International Conference on 2011 Jul 26 (pp. 3765-3769). IEEE.
71. Ferrer J, Kruse PM, Chicano F, Alba E. Evolutionary algorithm for prioritized pairwise test data generation. InProceedings of the 14th annual conference on Genetic and evolutionary computation 2012 Jul 7 (pp. 1213-1220). ACM.
72. Maragathavalli P, Kanmani S, Kirubakar JS, Sriraghavendrar P, Prasad AS. Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths coverage. InAdvances in Engineering, Science and Management (ICAESM), 2012 International Conference on 2012 Mar 30 (pp. 349-353). IEEE.
73. Mahajan M, Kumar S, Porwal R. Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach. ACM SIGSOFT Software Engineering Notes. 2012 Sep 7;37(5):1-5.
74. Singhal A, Chandna S, Bansal A. Optimization of Test Cases Using Genetic Algorithm 1.
75. Liu D, Wang X, Wang J. Automatic Test Case Generation based on Genetic Algorithm. Journal of Theoretical & Applied Information Technology. 2013 Feb 10;48(1).
76. Varshney S, Mehrotra M. Automated software test data generation for data flow dependencies using genetic algorithm. International Journal. 2014 Feb;4(2).
77. Garg D, Garg P. Basis Path Testing Using SGA & HGA with ExLB Fitness Function. Procedia Computer Science. 2015 Jan 1;70:593-602.
78. Khan R, Amjad M, Srivastava AK. Optimization of automatic generated test cases for path testing using genetic algorithm. InComputational Intelligence & Communication Technology (CICT), 2016 Second International Conference on 2016 Feb 12 (pp. 32-36). IEEE.
79. Bao X, Xiong Z, Zhang N, Qian J, Wu B, Zhang W. Path-oriented test cases generation based adaptive genetic algorithm. PloS one. 2017 Nov 14;12(11):e0187471.
80. Windisch A, Wappler S, Wegener J. Applying particle swarm optimization to software testing. InProceedings of the 9th annual conference on Genetic and evolutionary computation 2007 Jul 7 (pp. 1121-1128). ACM.
81. Zhu XM, Yang XF. Software test data generation automatically based on improved adaptive particle swarm optimizer. InComputational and Information Sciences (ICCIS), 2010 International Conference on 2010 Dec 17 (pp. 1300-1303). IEEE.
82. Cui H, Chen L, Zhu B, Kuang H. An efficient automated test data generation method. InMeasuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on 2010 Mar 13 (Vol. 1, pp. 453-456). IEEE.

83. Singla S, Kumar D, Rai HM, Singla P. A hybrid PSO approach to automate test data generation for data flow coverage with dominance concepts. International journal of advanced science and technology. 2011 Dec;37:15-26.
84. de Souza LS, de Miranda PB, Prudencio RB, Barros FD. A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort. InTools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on 2011 Nov 7 (pp. 245-252). IEEE.
85. Tiwari S, Mishra KK, Misra AK. Test case generation for modified code using a variant of particle swarm optimization (PSO) algorithm. In2013 Tenth International Conference on Information Technology: New Generations (ITNG) 2013 Apr 1 (pp. 363-368). IEEE.
86. Huang M, Zhang C, Liang X. Software test cases generation based on improved particle swarm optimization. InInformation Technology and Electronic Commerce (ICITEC), 2014 2nd International Conference on 2014 Dec 20 (pp. 52-55). IEEE.
87. Jiang S, Shi J, Zhang Y, Han H. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm. Neurocomputing. 2015 Jun 22;158:109-16.
88. Ting WY, Hui WD. Particle swarm optimization algorithm for test case automatic generation based on clustering thought. InCyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on 2015 Jun 8 (pp. 1479-1485). IEEE.
89. Jatana N, Suri B, Misra S, Kumar P, Choudhury AR. Particle swarm based evolution and generation of test data using mutation testing. InInternational Conference on Computational Science and its Applications 2016 Jul 4 (pp. 585-594). Springer, Cham.
90. Kumar S, Yadav DK, Khan DA. An accelerating PSO algorithm based test data generator for data-flow dependencies using dominance concepts. International Journal of System Assurance Engineering and Management. 2017 Nov 1;8(2):1534-52.
91. Li H, Lam CP. Software Test Data Generation using Ant Colony Optimization. InInternational conference on computational intelligence 2004 (pp. 1-4).
92. Farah R, Harmanani HM. An Ant Colony Optimization approach for test pattern generation. InElectrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on 2008 May 4 (pp. 001397-001402). IEEE.
93. Chen X, Gu Q, Zhang X, Chen D. Building prioritized pairwise interaction test suites with ant colony optimization. In2009 Ninth International Conference on Quality Software 2009 Aug 24 (pp. 347-352). IEEE.
94. Srivastava PR. Structured testing using ant colony optimization. InProceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia 2010 Dec 27 (pp. 203-207). ACM.
95. Bauersfeld S, Wegener J. A metaheuristic approach to automatic test case generation for GUI-based applications.
96. Bajaj, N., Khera, M., Vikram, Chahar, V. An Efficient Technique for Software Coverage using Metaheuristic Algorithm. International Journal of Computer Applications (IJCA) Proceedings on National Workshop-Cum-Conference on Recent Trends in Mathematics and Computing 2011, 14, 29 – 31
97. Mao C, Yu X, Chen J, Chen J. Generating test data for structural testing based on ant colony optimization. InQuality Software (QSIC), 2012 12th International Conference on 2012 Aug 27 (pp. 98-101). IEEE.
98. Alzubaidy LM, Alhafid BS. Proposed Software Testing Using Intelligent techniques (Intelligent Water Drop (IWD) and Ant Colony Optimization Algorithm (ACO)). International Journal of Computer Science Issues (IJCSI). 2013 Sep 1;10(5):91.
99. Yang S, Man T, Xu J. Improved ant algorithms for software testing cases generation. The Scientific World Journal. 2014;2014.
100. Arifiani S, Rochimah S. Generating test data using ant Colony Optimization (ACO) algorithm and UML state machine diagram in gray box testing approach. InTechnology of Information and Communication (ISemantic), International Seminar on Application for 2016 Aug 5 (pp. 217-222). IEEE.
101. Mala DJ, Mohan V. ABC tester-artificial bee colony based software test suite optimization approach. International Journal of Software Engineering. 2009 Jul;2(2):15-43.
102. Dahiya SS, Chhabra JK, Kumar S. Application of artificial bee colony algorithm to software testing. InSoftware Engineering Conference (ASWEC), 2010 21st Australian 2010 Apr 6 (pp. 149-154). IEEE.
103. Lam SS, Raju MH, Ch S, Srivastav PR. Automated generation of independent paths and test suite optimization using artificial bee colony. Procedia Engineering. 2012 Jan 1;30:191-200.
104. Singh T, Sandhu MK. An Approach in the software testing environment using artificial bee colony (ABC) optimization. International Journal of Computer Applications. 2012 Jan 1;58(21).
105. Suri B, Mangal I, Srivastava V. Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm. Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT). 2011 Jan:2231-5292.
106. Dalal S, Chhillar RS. A novel technique for generation of test cases based on bee colony optimization and modified genetic algorithm (BCO-mGA). Int. J. Comput. Appl. 2013 Apr 18;68(19):0975-8887.
107. Kumar S, Yadav DK, Khan DA. Artificial bee colony based test data generation for data-flow

testing. Indian Journal of Science and Technology. 2016 Oct 24;9(39).
108. Srivastava PR, Khandelwal R, Khandelwal S, Kumar S, Ranganatha SS. Automated test data generation using cuckoo search and tabu search (CSTS) algorithm.
109. Nasser AB, Sariera YA, Alsewari AR, Zamli KZ. A Cuckoo Search Based Pairwise Strategy For Combinatorial Testing Problem. Journal of Theoretical & Applied Information Technology. 2015 Dec 10;82(1).
110. Ahmed BS. Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. Engineering Science and Technology, an International Journal. 2016 Jun 1;19(2):737-53.
111. Alsewari AR, Zamli KZ. A harmony search based pairwise sampling strategy for combinatorial testing. International Journal of Physical Sciences. 2012 Feb 9;7(7):1062-72.
112. Xiang LY, Alsewari AA, Zamli KZ. Pairwise test suite generator tool based on harmony search algorithm (HS-PTSGT). NNGT Int. J. Artif. Intell. 2015 Feb;2:62-5.
113. Alsariera YA, Majid MA, Zamli KZ. A Bat-inspired strategy for pairwise testing. ARPN Journal of Engineering and Applied Sciences. 2015;10:8500-6.
114. Öztürk MM. A bat-inspired algorithm for prioritizing test cases. Vietnam Journal of Computer Science. 2018 Feb 1;5(1):45-57.
115. Nasser AB, Hujainah F, Alsewari AA, Zamli KZ. Sequence and sequence-less T-way test suite generation strategy based on flower pollination algorithm. InResearch and Development (SCOReD), 2015 IEEE Student Conference on 2015 Dec 13 (pp. 676-680). IEEE.
116. Nasser AB, Alsewari AA, Tairan NM, Zamli KZ. Pairwise test data generation based on flower pollination algorithm. Malaysian Journal of Computer Science. 2017 Sep 23;30(3):242-57.
117. Srivatsava PR, Mallikarjun B, Yang XS. Optimal test sequence generation using firefly algorithm. Swarm and Evolutionary Computation. 2013 Feb 1;8:44-53.
118. Sahoo RK, Mohapatra DP, Patra MR. A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases. International Journal of Computer Sciences and Engineering. 2016;4(8):54-8.
119. Zakaria HL, Zamli KZ. Migrating Birds Optimization based strategies for Pairwise testing. InSoftware Engineering Conference (MySEC), 2015 9th Malaysian 2015 Dec 16 (pp. 19-24). IEEE.
120. Sahoo RK, Ojha D, Mohapatra DP, Patra MR. Automated Test case Generation and optimization: A Comparative Review. International Journal of Computer Science & Information Technology. 2016;8(5):19-32.
121. Khurana N, Chhillar Rs. A Comparison Of Evolutionary Techniques For Test Case Generation And Optimization. Journal of Theoretical & Applied Information Technology. 2017 Oct 15;95(19).
122. Wang L, Chen K. Advances in Natural Computation: Pt. 1: First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings. Springer Science & Business Media; 2005 Aug 17.
123. Tian DP, Li NQ. Fuzzy particle swarm optimization algorithm. InArtificial Intelligence, 2009. JCAI'09. International Joint Conference on 2009 Apr 25 (pp. 263-267). IEEE.

تحليل حول قابلية تطبيق تقنيات البحث الفوقية على إنشاء بيانات الاختبار الآلي في تقييم البرمجة التلقائي

جعفرو موسى **روهيدا رومل** **نوريانى يوسف**

مدرسة الحوسبة ، جامعة أوتارا ماليزيا ، ماليزيا

الخلاصة:

حظي تقييم البرمجة التلقائي (APA) بالكثير من الاهتمام بين الباحثين بشكل أساسي لدعم الدرجات الآلية ووضع علامات على المهام المكلف بادائها الطلاب أو التدريبات بشكل منهجي. يتم تعريف APA بشكل شائع كطريقة يمكن أن تعزز الدقة والكفاءة والاتساق وكذلك تقديم ملاحظات فورية لحلول للطلاب. في تحقيق APA ، تعد عملية إنشاء بيانات الاختبار مهمة للغاية وذلك لإجراء اختبار ديناميكي لمهمة الطلاب. في مجال اختبار البرمجيات ، أوضحت العديد من الأبحاث التي تركز على توليد بيانات الاختبار نجاح اعتماد تقنيات البحث الفوقية (MHST)من أجل تعزيز إجراءات استنباط بيانات الاختبار المناسبة للاختبار الفعال. ومع ذلك، فإن الأبحاث التي أجريت على APA حتى الآن لم تستغل بعد التقنيات المفيدة لتشمل تغطية اختبار جودة برنامج أفضل. لذلك ، أجرت هذه الدراسة تقييماً مقارناً لتحديد أي تقنية بحث فوقي قابلة للتطبيق لدعم كفاءة توليد بيانات الاختبار الآلي (ATDG) في تنفيذ اختبار وظيفي ديناميكي. في تقييم البرمجة التلقائي يتم تضمين العديد من تقنيات البحث الفوقية الحديثة في التقييم المقارن الذي يجمع بين كل من خوارزميات البحث المحلية والعالمية من عام 2000 حتى عام 2018.

تشير نتيجة هذه الدراسة إلى أن تهجين Cuckoo Search مع Tabu Search و lévy flight كواحدة من طرق البحث الفوقية الواعدة ليتم تطبيقها ، حيث أنه يتفوق على الطرق الفوقية الأخرى فيما يتعلق بعدد التكرارات ونطاق المدخلات.

الكلمات المفتاحية: تقييم البرمجة التلقائي ، تقنيات البحث الفوقية ، توليد بيانات الاختبار الآلي