Journal of Advances in Mathematics and Computer Science

35(4): 115-134, 2020; Article no.JAMCS.58337
ISSN: 2456-9968
(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

Discrete Artificial Dragonflies Algorithm in Agent Based Modelling for Exact Boolean *k*Satisfiability Problem

Hamza Abubakar[1*], Sagir Abdu M.[2], Surajo Yusuf[3] and Yusuf Abdurrahman[3]

[1]School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia.
[2]Department of Mathematics, Federal University, Dutsin-Ma, Katsina State, Nigeria.
[3]Department of Mathematics, Isa Kaita College of Education, Dutsin-Ma, Katsina State, Nigeria.

Authors' contributions

This work was carried out in collaboration among all authors. Author HA designed the study, performed the statistical analysis, wrote the protocol and wrote the first draft of the manuscript. Authors SAM and SY managed the analyses of the study. Author YA managed the literature searches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JAMCS/2020/v35i430275
Editor(s):
(1) Dr. Dariusz Jacek Jakóbczak, Koszalin University of Technology, Poland.
Reviewers:
(1) C. Natarajan, Anna University, India.
(2) R. M. Kapila Tharanga Rathnayaka, Sabaragamuwa University of Sri Lanka, Sri Lanka.
Complete Peer review History: http://www.sdiarticle4.com/review-history/58337

Original Research Article

Received: 20 April 2020
Accepted: 25 June 2020
Published: 07 July 2020

Abstract

The development of metaheuristics and Boolean Satisfiability representation plays an important part in a neural network (NN) and Artificial Intelligence (AI) communities. In this paper, a new hybrid discrete version of the artificial dragonfly algorithm (DADA) applying a minimum objective function in agent-based modelling (ABM) obeying a specified procedure to optimize the states of neurons, for optimal Boolean Exact Satisfiability representation on NETLOGO as a dynamic platform. We combined the artificial dragonfly algorithm for its random searching ability that encourages diverse solutions and formation of static swarm's mechanism to stimulus computational problems to converge to the best global optimal search space. The global performance of the proposed DADA was compared with genetic algorithm (GA) that are available in the literature based on the global minimum ratio (gM), Local Minimum Ratio (yM), Computational time (CPU) and Hamming distance (HD). The final results showed good agreement between the proposed DADA and discrete version of GA to efficiently optimize the Exact-*k*SAT problem. It found that DADA-ABM has high potentiality for optimizing or modelling a network that is very hard or often impossible to capture by exact or traditional optimization modelling techniques such as Boolean satisfiability problem is better than existing methods in the literature.

**Corresponding author: E-mail: zeeham4u2c@yahoo.com;*

Keywords: *Logic program; artificial dragonfly algorithm; Boolean Satisfiability Problem (SAT); exact ksatisfiability; agent-based modelling.*

1 Introduction

The Satisfiability of Boolean formulas in Conjunctive Normal Form (SAT Problem) is a central problem in theoretical computation and discreet mathematics. In optimization terminology, propositional Satisfiability is a combinatorial optimization problem (COP) in a discrete domain and can be graded as an NP-hard in most of its variants. This type of problem cannot be easily solved by analytical or calculus-based optimization methods. Indeed, it needs a powerful method that can function dynamically to find the right solution [1]. There are several problems in various fields that are known to be combinatorial optimization in nature that are very difficult to tackle through analytical or exact based methods that are based on the concept of listing all solutions as it takes a long time, particularly if the size of the problem is increasing [2-3]. SAT is certainly one of the most common problems in combinatorial optimization; it belongs to graph theory, planning and scheduling, sports, databases, circuit design, and the artificial intelligence family. The idea of the Satisfying of the Boolean Formula is to decide if the Boolean Formula in the Conjunctive Normal Form (CNF) has a true assignment that satisfies each clause or decides that no label assignment exists [4]. Various optimization problem can be translated into Exact *k*SAT representation is closely related to several interesting NP-complete problems, such as Exact Hitting Set [5], minimum hitting set [6] and Exact cover [7], monotone ExactSAT is the same as the exact hitting [8], set problem on hypergraphs [9], Exact graph colouring problem [10], exact independent set [11] and it is closely related to the set partitioning problem [12]. Various NP-complete problems such as N-queen problems, Scheduling problem, Transportation problem can be transformed and represented in form Exact*k*SAT and have many applications in combinatorial optimization that falls into the category of optimization, sorting, decision or counting scheme for solving the satisfiability problem and is sub-optimal and partially heuristic in nature.

Metaheuristic algorithms are becoming a common tool utilized by various researchers in the field of the combinatorial optimization problem for its ability to provide near-optimal solutions to difficult searching and other combinatorial optimization problem [13]. A wide range of metaheuristic optimization algorithms has been proposed by researchers in the field of optimization, these include, Artificial bee colony (ABC) [2], Ants Colony Optimization (ACO) [14], Genetic algorithm (GA) and Firefly algorithm (FFA) [15], Artificial immune system (AIS) [16] and other popular examples of nature-inspired metaheuristic algorithms that are popular in solving various optimization problem include the work of [17], Blum, et al. [18], Juan et al. 2015 [19], Essaid et al. [20], Salhi, et al. [21], Pei, et al. [22], Tessaro et al. [23], Liu [24]. Metaheuristics methods are one that; (i) search for an approximate solution, (ii) need not particularly have a mathematical convergence proof, and (iii) does not explore each possible solution in the search space before arriving at the final solution, hence is computationally efficient Bandaru, et al. [25].

Dragonfly Algorithm (DA) is one of the newly developed algorithms that rely on Swarm Intelligent (SI) is ADA, which is part of a population-based meta-heuristic algorithm. Dragonflies are amazing creatures found in nature more than 300 million years ago. It is one of the recently created metaheuristic algorithms that influenced the distinctive and superior scurrying behaviour of dragonflies in nature, such as navigating, minimizing the enemy of dragonflies, and maximizing food source. It was successfully and effectively examined and tested on 19 benchmark functions, as well as very statistical results compared to other excellently-known algorithms in the literature. DA has been successfully evaluated and tested on mathematical benchmark functions, as well as, it obtained very comparative results compared to other well-known algorithms [26]. Moreover, DA has been used to solve many real-world optimization problems such as Travelling Salesmen problem, feature selection problem [27], numerical optimization problems [28], Generation dispatch of combined solar thermal systems [29], distribution network reconfiguration [30], image segmentation [31] and distribution networks [32].

1.1 Agent-based modeling

ABM is a kind of computational model that examines the structures of multiple interacting agents that are spatially located and evolve. ABMs are extremely useful in demonstrating how complex patterns arise from micro-level laws over time. Unlike ABMs, which are based on deductive reasoning, ABMs function properly not only as an inductive reasoning methodology, where a conclusion is drawn from a series of observations but also as a pure form of adductive reasoning, in which the best explanation for the phenomena under research is extrapolated by simulation [33]. ABMs have had a profound influence in many areas of research, engineering and industry. However, to achieve their optimal potential, these models should be enhanced to meet any suitable optimization approaches that have so far been shoved aside or left unattended. This is because the exact method or conventional optimization procedures are unable to address due to non-linearity or complexity of the problems that arise. The needs for a simple and flexible model like the ABM to attracted support from the field of metaheuristics algorithm communities for optimal performance. The fact that even complicated patterns can be described by mathematical or analytical models based on a few basic decision rules is remarkable and such modelling is still a vibrant and important area of research [34]. The core concept behind agent-based modelling is to characterize a structure as a starting point, applying its constituent parts. ABMs are widely recognized for their effectiveness in the analysis of complex activities. The underlying structure of ABMs differs, but they are usually built to constitute autonomous software objects that communicate within the framework. Agents are often represented as having activities and tasks that may have an impact on the environment including on each other and are especially likely to be combined to create interactive models and simulations [35-36]. Abstract characterizations of ABMs, however, can be seen mainly as "after-the-fact" attempts to put together ideas that are intuitively communicated by agent terminology. While there may not be a well-defined and universally agreed definition of agent-based models, the relevance of ABMs in science and industry is evident in their diverse implementations. These include studies into disease transmission, the functioning of ecosystems, the dynamics of supply chains, fluctuations and developments in financial markets, the behaviour of economic sectors, patterns of migration, urban traffic patterns and interactions between chemical and biophysical processes.

Besides, the current role of optimization in agent-based modelling is entirely analogous to the function that is assumed in the general field of simulation just a few years ago as a result of the same factors – the inapplicability of classical models, non-linearity, combination complexity and uncertainty. In the simulation industry, practitioners have struggled for years to tackle the compelling issues of optimization simply employing a trial-and-error analysis. Simulation literature also claimed to have "optimized" various machine components based on nothing more than a set of conjectures and brute force re-trials.

ABM can typically be built through three iterative phases: (1) the design process, (2) the programming process and, eventually, (3) the evaluation and review phase. The initial skeleton of ABMs is constructed during the design process. This initial skeleton is essentially a textual ABM model in which all behavioural rules and properties of agents and the environment, along with how they communicate with each other, are verbally registered. As a second phase, the programming process deals with how to turn ABM's textual model into a computational model through agent-based programming languages and toolkits. The third step of the study and review shall be carried out for insight. However, we are not aware of any previous work that optimizes Exact *s*atisfiability of the Boolean formula using Artifcial Dragonflies algorithm in the concerning a group of individuals in an agent-based model using the Netlogo framework. We are involved in developing a new approach to modelling ABM to be able to catch essential neuron behaviours. The contributions of the present study include the following:

1. To reformulate ADA operators to deal with discrete optimization rather than continuous optimization.
2. To develop hybrid artificial dragonflies algorithm in Exact *k*SAT.
3. To implement newly proposed hybrid artificial dragonflies in Exact *k*SAT based on agent-based computational modelling Netlogo.
4. To explore the feasibility of mapping DADA in Exact *k*SAT term of global minimum ratio, local minimum ration, Hamming Distance and Computation time.

2 Exact *K*satisfiability Logical Representation of Boolean Formula

The Exact Satisfiability of a Boolean formula is considered as a decision problem which decided a Boolean formula in Conjunctive Normal form (CNF) has a truth assignment satisfying exactly one literal in each clause or determine that no such label assignment exists; The Exact *k*Satisfiability (Exact*k*SAT) problem is a variant of Satisfiability (SAT), where the input instance is the same but the question is that in Exact*k*SAT a clause is satisfied if *exactly* one of its literals is true (instead of at least one literal, as in ordinary *k*-SAT). The Exact *k*Satisfiability (Exact *k*SAT) problem is the variant of Exact *k*SAT in which each clause contains at most *k* literals. Exact3SAT is also called One-In-Three Satisfiability. Exact *k*SAT is NP-complete even when restricted to clauses containing at most three literals and all variables occurring only unnegated [12]. Exact *k*SAT with all variables occurring at most twice can be solved in polynomial time [37]. We focus on Exact *k*SAT or more precisely on Exact3SAT [4].

Consider a Boolean expression $Q_{ExactkSAT}$ that is built from Boolean variables in a conjunctive normal form which has the following properties.

i. Consisting of a set of Boolean variables, $\left(x_1, x_2, x_3, ..., x_n\right)$, where x_i is a variable or its negation $\neg x_i$
ii. A collection of literals. Literal is defined as a variable x_i or its negation;
iii. A collection of m distinct logical clauses $C_i \in \left(c_1, c_2, c_3, ..., c_m\right)$;
iv. Each satisfying assignment satisfies exactly one literal in each clause;
v. Each variable in the logical clause are linked by Boolean connectives OR;
vi. Each logical clause consists of literal linked by a Boolean connective AND ($\wedge$) ;
vii. Each clause C_i is a disjunction of exactly three literals;
viii. Each clause C_i contains at most three literals.

These properties simplify the formulation of the problem via a Hopfield neural network and preserve NP-completeness (Schaefer, 1978) [4]. The Boolean values for each x_i are bipolar $x_i \in \{-1,1\}$ that exemplifies the notion of FALSE and TRUE respectively. The general formulation of Exact *k*SAT is presented as follows:

$$Q_{ExactkSAT} = \bigwedge_{i=1}^{k} C_i \tag{1}$$

when $k = (1,2,3)$, Equation (1) describes the Boolean formula for Exact *k*SAT containing of logical clause C_i given in Equation (2) as follows:

$$C_i = \bigvee_{j=1}^{3} \left(E_{ij}, D_{ij}, F_{ij}\right) \tag{2}$$

where $E_{ij}, D_{ij}, F_{ij} \in [1,-1]$. If the context is clear we denote the number of clauses of some formula Exact*k*SAT [40]. The Boolean value for the mapping is expressed as 1 (TRUE) and -1 (FALSE).

Examples for Exact*k*SAT formulation for $k = 3$ is presented as follows based on [4].

$$Q_{ExactkSAT} = \left(E_1 \vee E_2 \vee E_3\right) \wedge \left(D_1 \vee \neg D_2 \vee D_3\right) \wedge \left(\neg F_1 \vee F_2 \vee F_3\right) \tag{3}$$

Equation (3) is satisfiable since it gives true value resulting in $Q_{ExaxtkSAT} = 1$. If the neuron states are considered as $E_i (i = 1,2,3)$, $D_i (i = 1,2,3)$ and $F_i (i = 1,2,3)$ the Boolean expression will be unsatisfiable if $Q_{EXACTkSAT} = -1$. The properties of DADA can be used to govern the behaviour of Exact *k*SAT formulation. Indeed, it requires a strong method that can behave in a dynamic way to find a proper solution. To the knowledge of the author, discrete artificial dragonfly algorithm (DADA) has not been used to attack the Exact satisfiability problem on Agent-Based Modelling yet.

3 Proposed of DADA for Exact *K*sat

The objective of Exact Satisfaction (Exact SAT) is to establish if a Boolean Formula (BF) in Conjunctive Normal Form (CNF) has a true assignment that satisfies exactly one literal in each clause or decides that no label assignment exists. Since the original version of artificial dragonfly algorithm (ADA) is working on a continuous optimization problem, the SAT problem is considered to be discrete. Thus, in this study, the ADA will be modified to tackle the problems of Exact *k*SAT. The following steps explain the technique used in the proposed method. One of the goals of utilization is to reformulate ADA operators to deal with discrete optimization rather than continuous optimization.

This limitation involves an algorithm that can effectively turn (update) the neuron state based on a previously improved solution with diverse solution space. In general, the value of the fitness function of the variable d_r^f in the ADA obeys the following;

$$\lambda_{Q_{ExactkSAT}} = d_1^f(x) \; + \; d_2^f(x) \; + \; d_3^f(x) \; +,...,d_{N_{Dragonflies}}^f(x) = \sum_{i=0}^{m} C_i^{(k)} \tag{4}$$

where $f \in [1,2,3...N]$ and d_r^f designates to the location of the *r*th dragonfly in the *d*th dimensional space and $N_{dragonflies}$ is the number of potential search agents.

where $\lambda_{Q_{ExactkSAT}}$ is the maximum number of satisfied clauses, *m* describes the maximum number of a clause in $Q_{ExactkSAT}$ the program and $C_i^{(k)}$ is the clauses tested by the DADA given as follows;

$$C_i^{(k)} = \begin{cases} 1 & , \; Satisfied \\ 0 & , \; otherwise \end{cases} \tag{5}$$

Each neuron string represents the assignment that corresponds Exact *k*SAT instances. The objective function of our proposed ADA is to maximize the fitness of the artificial dragonflies d_r^f (neuron string). In general, global optimization can be presented in Equation (6) without the loss of generality in a minimization problem.

$$\max\left[\lambda_{Q_{ExactkSAT}} = \sum_{i=0}^{m} C_i^{(k)}\right] \tag{6}$$

The mapping of ADA for Exact *k*SAT is abbreviated as DADA-Exact*k*SAT. The stages involved in DADA-Exact*k*SAT are presented as follows:

3.1 Stage 1: Initialization

The goal of any optimization is to search for the best solution in terms of the variables of the problem. An array of vector variable to be optimized is formed. An initial population of the size $N_{Dragonflies}$ of Artificial Dragonflies $d_r^f = \left[d_1^f, d_2^f, d_3^f, ..., d_{N_{Dragonflies}}^f \right]^T$ for each solution and step matrix Δd_r^f were generated. The state of each Artificial Dragonflies d_r^f in the search space is denoted by 1 or -1 which represent the *True* or *Falsification* that corresponds to the possible mapping for Exact *k*Satisfiability problem. Every solution is randomized within the variable boundaries range based on Equation (7),

$$d_r^f = \lambda_{d_r^f}^{\min} + d_1^f * \left(\lambda_{d_r^f}^{\max} - \lambda_{d_r^f}^{\min} \right) \tag{8}$$

where $r \in [1,2,3...N_{Dragonflies}]$ and d_r^f designates to the location of the *r*th dragonfly in the *f*th dimensional space and $N_{Dragonflies}$ refers to the number of potential search agents. A uniformly distributed random is defined as $d_1^f \in [0,1]$. This problem searches for the optimal Exact *k*Satisfiability.

3.2 Step 2. Calculate the distance for each artificial dragonfly

The distance from the neighbourhood is determined by computing and selecting the Euclidean distance all the dragonflies and picking *N* of them. The distance σ_{ij} is to determine obeys Equation (9),

$$\sigma_{ij} = dist\left(d_{ir}^f, d_{jr}^f \right) \tag{9}$$

Where d_{ir}^f and d_{jr}^f are the fitness of the neighbourhood areas and artificial dragonfly respectively. The artificial dragonfly will be assigned to its neighbourhood areas based on their fitness values.

3.3 Stage 3: Fitness evaluation

The variable vector is tested based on the fitness $\lambda_{Q_{ExactkSAT}}$ to a quantified variable position in the initialized solution space. All randomized variable vector will undergo fitness function assessment. The fitness $\lambda_{d_r^f}$ of each d_r^f is computed based on the initial position variables that are generated randomly between the lower and upper limits of the variables using Equations (4) and (5). The parameters of each artificial dragonfly are the same as variables in the optimization problem. The combination of parameters determines the attractiveness of artificial dragonfly. In this case, the proposed model obeys $\lambda_{d_r^f} \in N_{Dragonflies}$. The maximum fitness of the d_r^f is $\lambda_{d_r^f}^{\max} = \lambda_{d_r^f}^{\min}$ and if $\lambda_{d_r^f}$ the algorithm will be terminated when the optimum fitness is reached.

3.4 Stage 4: Update the position and velocity of artificial dragonflies

The position and velocity of dragonflies' coefficients of $S_{d_r}^f$, $A_{d_r}^f$ and $C_{d_r}^f$ are determined and updated (flip) based on the following equations:

3.5 Separation strategy $\left(S_r^f\right)$

This process states the avoidance of static collisions between artificial dragonflies in the same neighborhood.

$$S_{d_r}^f = -\sum_{d_r=1}^{N} d_r^f - d_i^f \tag{10}$$

3.6 Alignment strategy $\left(A_r^f\right)$

This refers to the pace at which artificial dragonflies play in the same neighbourhood.

$$A_{d_r}^f = \frac{1}{N}\sum_{d_r=1}^{N} \Delta\Psi_i^f \tag{11}$$

3.7 Cohesion strategy $\left(C_r^f\right)$

This technique demonstrates the artificial dragonfly's inclination toward the middle of the neighborhood. Preserve the unity of the artificial dragonflies in setting the path to the neighborhood core.

$$C_{d_r}^f = \left(\frac{1}{N}\sum_{d_r=1}^{N} d_r^f\right) - d_i^f \tag{12}$$

where d_r^f and Ψ_r^f describe the position and velocity of the rth potential dragonfly and d_i^f described the position of the current artificial dragonfly and N is the number of neighbouring potential search agent.

3.8 Stage 5: Update food $\left(F_r^f\right)$source and enemy source$\left(E_r^f\right)$

The attractiveness of artificial dragonfly towards food source and distraction from its enemy obey the following equations;

$$F_{d_r}^f = \zeta_{food} - d_i^f \tag{13}$$

$$E_{d_r}^f = \zeta_{ememy} + d_i^f \tag{14}$$

where d_i^f describes the position of the current artificial dragonfly and ζ_{food} ζ_{ememy} describe the food source and the enemy source respectively. The artificial dragonflies sources for food and enemy are expressed as the best and worst solutions respectively observed so far in the solution space.

The adaptive integration of the preceding operations helps to correctly flip the artificial dragonfly position. This applies phase vector and position vector to swap the position of artificial dragonflies in a search space and to simulate their hunting movements every round. The phase vector indicates the swapping direction of the artificial dragonflies as follows.

$$\Delta d_r^f = \left(sS_{d_r}^f(t) + a_r A_{d_r}^f(t) + c_r C_{d_r}^f(t) + f_r F_{d_r}^f(t) + e_r E_{d_r}^f(t) \right) + w_r \Delta d_r^f(t) \tag{15}$$

Where w_r defined the *inertia weight* of potential search agent, s_r, a_r, c_r, f_r, and e_r are the weights of separation, alignment, cohesion, food attraction, enemy distraction respectively.

In the discrete domain, the position of the artificial dragonflies can be updated obeying the transfer function in which the velocity values are obtained as inputs and number in d_1^f which represents the likelihood of switching artificial dragonfly positions in the solution space [26,38].

In the DADA it is seen that an artificial dragonfly moves by flipping the number of bits. Consequently, the artificial dragonfly's velocity may be represented by changing the probabilities of bit changed per iteration. In other words, an artificial dragonfly moves within a search space by only assuming values of -1 or 1, where each velocity represents the probability of a bit of position taking the value 1. To be within the range [0,1], the velocity, which is a probability, must be limited [39]. A function used to do this is called the sigmoid function, and is formulated mathematically as follows:

$$T(d_r^f) = \frac{1}{1 + \exp(-d_r^f)} \tag{16}$$

After calculating the probability of changing position for all dragonflies, Equation (17) is employed to update (flip the neurons) the position of search agent in bipolar search spaces. The location switch is defined by comparing with the uniformly generated random numbers between 0 and 1 which are formulated as follows:

$$d_r^f(r+1) = \begin{cases} 1 & rand[0,1] > T(d_{r+1}^f) \\ -1 & otherwise \end{cases} \tag{17}$$

With the above components, the DADA can repeatedly flip the bipolar bits in solutions until the satisfaction of an end condition.

3.9 Stage 6: New solution generation: Finally, the emergence of a new population of bipolar artificial dragonflies.

Go back to Phase 2, quantify the cost of each artificial dragonfly within the new population and then perform the loop until the stop condition is met. The fitness function is then calculated obeys the updated location and velocities. The modification of artificial dragonflies' location continues until the criteria are met.

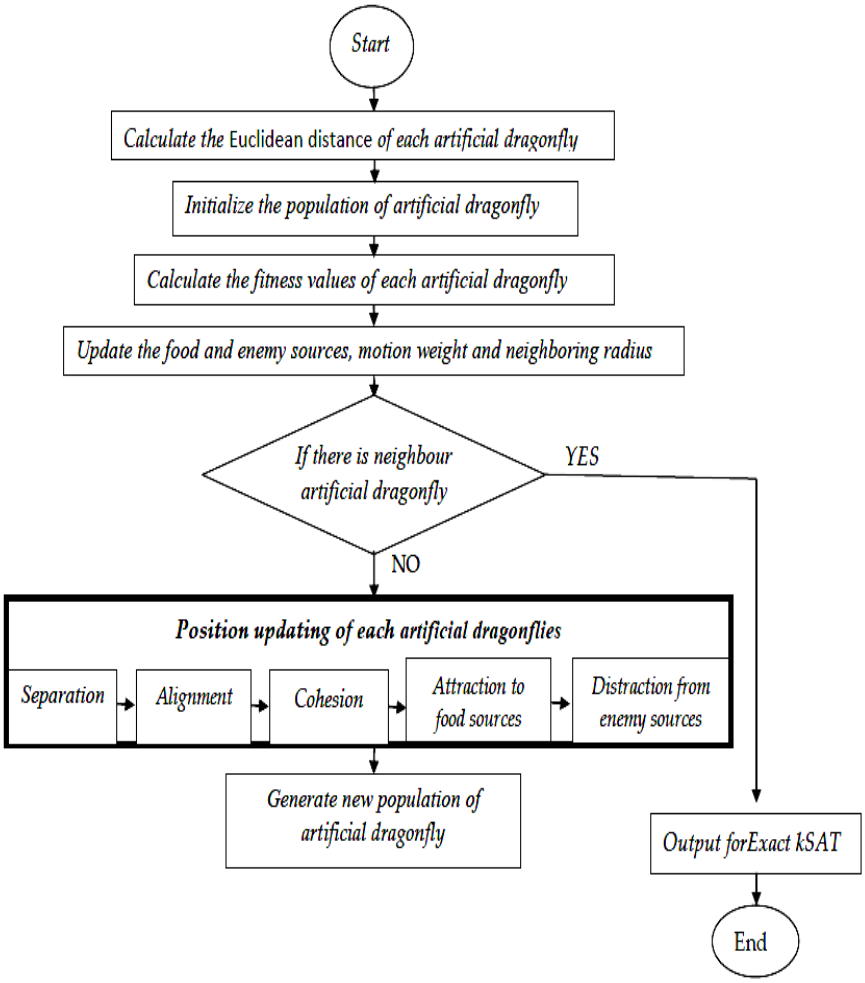

Fig. 1. Flowchart of artificial dragonfly algorithm

4 DADA-based for Exact *k*Satisfiability Problem

As mentioned in Section 2, the objective of the Exact *k*Satisfiability of a Boolean formula is considered as a decision problem which decided a BF in CNF has a truth assignment satisfying exactly one literal in each clause or determine that no such label assignment exists. Since the original version of dragonfly algorithm worked for continuous optimization problems, and the Exact *k*Satisfiability problem is considered as a discrete optimization problem. Thus, in this research, DADA will be adapted for tackling Satisfiability. The following steps illustrate the used procedure in the proposed approach:

1. Create a set of initial solutions for DADA obeying Equation (8).
2. Calculate the value of the fitness function of the variable obeying equation (4) and (5).
3. Calculate the overall distance for each solution obeying equation (9).
4. Determine the maximum number of iterations obeying equation (6).
5. Find the consistency of all solutions to spread solutions across different neighborhoods area based on equation (5). If the similarity rate (the number of similar bits divided by the total number of bits in the solution space) is greater than a predefined value between two neighbouring solutions, then they will be

classified as two neighbourhood in the solution space. In Fig. 2, there are two solutions S_1^i and S_2^i which has been presented as a vector of $i \in 1,2...N$ variables, each element represents a possible representation to Exact *k*SAT logic. The index of an element represents the order of bits in the solution space. A similarity index is a matrix that contains the value 1 in the index i if the bits in the index i are both S_1^i and S_2^i are satisfiable, otherwise, the similarity index will have the value 0 obeying Equation (5).

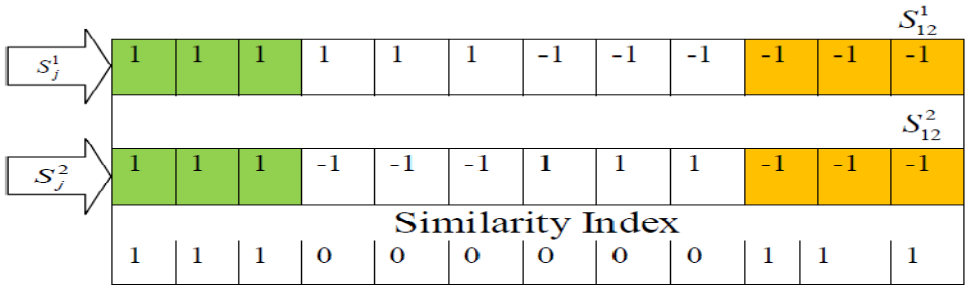

Fig. 2. The solution representation and similarity index of DADA

6. Determine the solutions that represent the food and enemy source in the population using Eqs (13) - (14).
7. Adjust the DADA weights *s, a, c, f, e* and *w*.
8. Define the best artificial dragonfly in each neighbourhood in the solution space.
9. Execute the alignment, separation, cohesion, food attraction, and enemy distraction operations obeying Equations (10)-(12).
10. Update the bits position the emergence of a new population of bipolar artificial dragonflies based on Equation (16) and (17).

4.1 Separation mechanism

The diversification of the S_j^i location will take place at this stage separation involves the exchange of two substructures in two-bit regions S_j^i. The location of the separation strategy shall be determined at random. If the similarity rate between two potential solutions (located in the same neighbourhood in the solution space) exceeds the predefined value, then two artificial dragonflies (bits) that exist in the same elements in both solutions are randomly selected and exchanged. This strategy will be repeated between the current solution and all its neighbouring solutions in the solution space. Fig. 3, illustrates an example of a DADA separation mechanism between two neighbourhoods S_1^i and S_2^i.

4.2 Alignment mechanism

Alignment operator comprises state flipping from 1 to -1 or vice versa. This mechanism will theoretically improve the average fitness of S_j^i. It is executed by winding up the current solution to the best solution in the neighbourhood, as shown in Fig. 4, S_j^1 and S_j^2 obtain the best solution and the current solution in the current neighbourhood in the solution space. Two consecutive Artificial dragonflies location are picked randomly from the best solutions obtained (i.e., S_j^1 and S_j^2). If the S_j^1 and S_j^2 in the solution space are

not in the same order in the current solution (i.e., S_j^2), then the alignment mechanism is executed by swapping the S_2^2 with the next Artificial dragonfly (bit) location of S_1^i in the solution space.

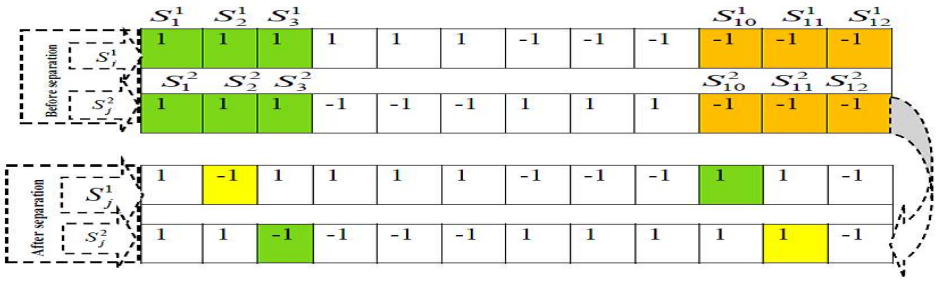

Fig. 3. Shows the separation mechanism of DADA

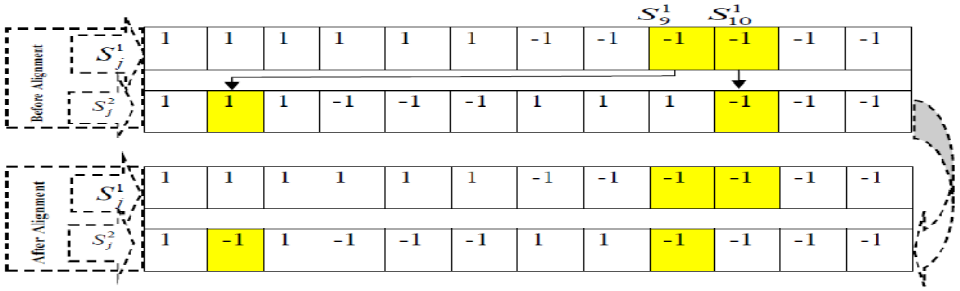

Fig. 4. Shows alignment mechanism in DADA

4.3 Cohesion mechanism

This mechanism allow for rounded together all solutions to preserve the mass of a neighbourhood within the same neighbourhood in the solution space. The principle of cohesiveness is close to that of the alignment mechanism, except that only the current and the optimal solutions for the current latest are executed.

4.4 Food attraction operation

This is another variant of the alignment mechanism. The current solution is rounded off to the best solution in the population during the food attraction process (which constitutes the food source within the AD).

4.5 Enemy distraction operation

The present solution in the population is steered separately from the worst solution. Fig. 5 demonstrates the concept of the operation. S_j^1 and S_j^2 described the worst and the current solution in the solution space. Enemy distraction operation tries to find two consecutive artificial dragonflies (variables) that exist in the same order in both S_j^1 and S_j^2. If the S_1^1 and S_2^1 are found, The enemy diversion is then implemented by swapping S_2^1 with a random variable in S_2^2.

11. Repeat the steps 4 to 12 until the stopping requirement is fulfilled.
12. Return the best solution found.

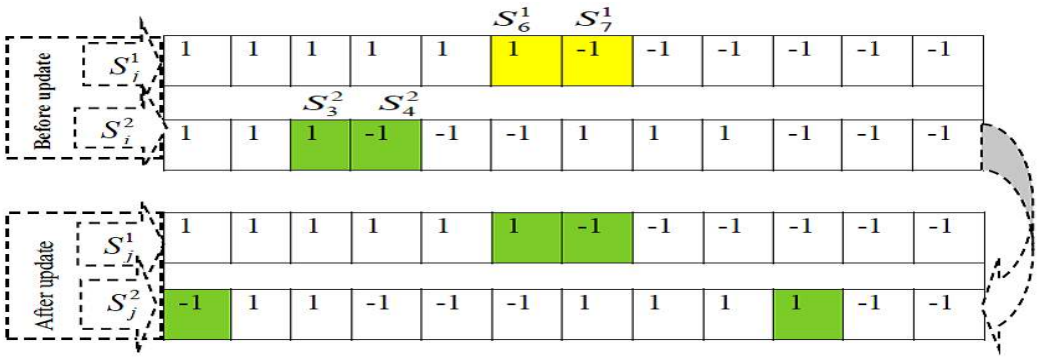

Fig. 5. Enemy distraction mechnism in DADA

5 Methodology Agent-based Modeling for Dada-exactsat

Stage 1: Entering the value of Each Parameters

1. Press the setup button to initialize the model and solution space button for the new user. Users will click the next button to take the next step and the last step.
2. In this step enter the value of number of neurons involved *NN*, and the clauses numbers NC1, NC2, NC3, and setups the value of Relax μ, value of COMBMAX.
3. In this step we should choose type of activation function and learning.
4. Next, slides the slider to fix the number of trial, number of learning events while the maximum of tolerance.
5. After setting the values, press the setup button in the program and press set.
6. Finally, press the button "go" to run the program.

Stage 2: Training phase

7. Initialize initial states of neuron in each clauses. Based on the design of DADA that stimulate the behaviors of neurons update, all neuron will communicate with each other in a complete bonded form as all of them tries to reach appropriate neural state.
8. Computing final state for the corresponding satisfiable state of Exact *k*SAT logic.
10. Accept the final solution as a global solution if the Exact *k* SAT solution remain unchanged after five runs, or else go back to step 1.
11. Finally, the global minimum solution, the Local minimum solution, computational time and Hamming distance are generated.

6 Experimental Setup for DADA-Exact*k* SAT Models

The algorithms used in this work were programmed using ABM with Netlogo programming language on a personal computer with an Intel dual-core processor running at 2 GHz. In order to investigate the performance of DADA for solving the Exact *k*SAT. The proposed approach was implemented on ABM using NETLOGO as a platform Intel ® Core™ i7 8th generation CPU with 8 GB of RAM. The control parameters of the DADA based approach for Exact *k*SAT representation were fixed for all simulated results generated in Table 1. The performances of the proposed hybrid model was compared with GA performance that available in the literature.

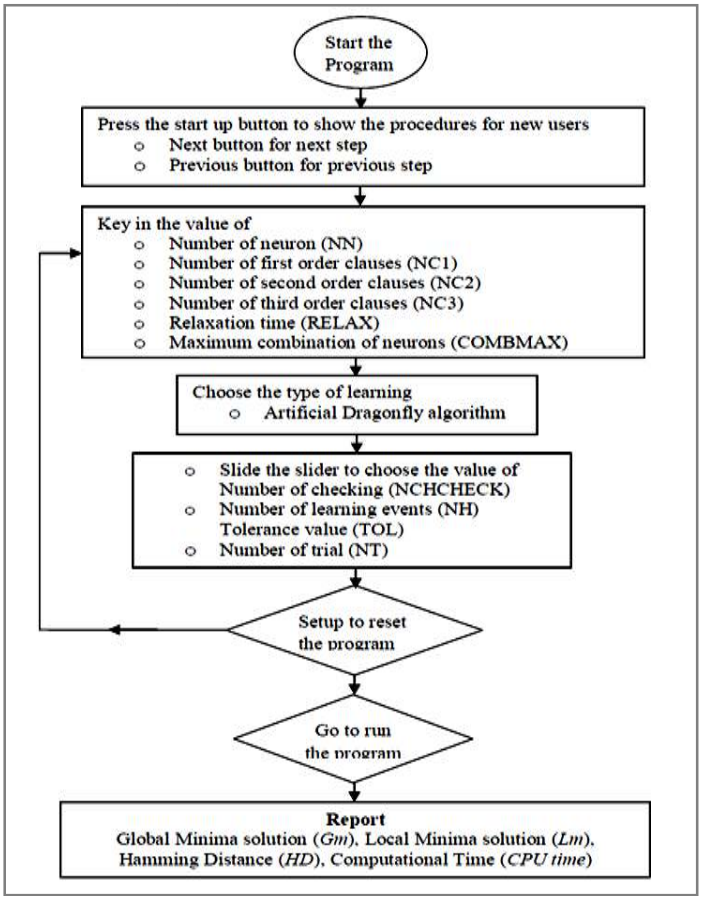

Fig. 6. Methodology of agent-based modeling for DADA exact SAT

Table 1. List of parameters for the DADA algorithm

Parameter	Value
Neuron Combination	100
Number of Trials	100
Population_size	100
Selection_Rate	0.5
Separation operator	0.2
Alignment operator	0.2
Cohesion operator	0.5
Food Attraction operator	1.0
Enemy Distraction operator	1.0
Number of iteration	10,000

7 Evaluation on Performance

The evaluation of performance is a key aspect of the design process of a hybrid metaheuristics model. It is deemed that a sufficiently reliable estimate of accuracy and precision of predictions of a model is given by measurements made on the differences between the expected values and those observed, said "difference

measures". Once the training process finished, the neural network calculated the values for the Global Minimum ratio (gM), local Minimum Ratio (yM), Hamming Distance (HD) and Computation time (CPU),

7.1 Global minimum ratio

$$gM = \frac{1}{n}\sum_{i=1}^{n} H_{\square\ ExactkSAT}$$

7.2 Local minimum ratio

$$yM = \frac{1}{n}\sum_{i=1}^{n} (H_{\square\ ExactkSAT} - H^{\min}_{\square\ ExactkSAT})$$

7.3 Hamming distance

$$HD = \frac{1}{tc}\sum_{i=1}^{k} \left| S_i(t+\Delta t) - S_i(t) \right|$$

7.4 Computational time

CT_Time = Learning_Time + Retrieval_Time

8. Results and Discussion

The outcomes are reported after simulation of the DADA-Exact *k*SAT logical representation was executed system using various performance evaluation metrics. The *gM*, yM, *HD* and *CPU time* of the models are plotted and presented in Figs. 7-10. The objective is to review the performance of DADA and GA in doing Exact *k*SAT logic programming via simulated data.

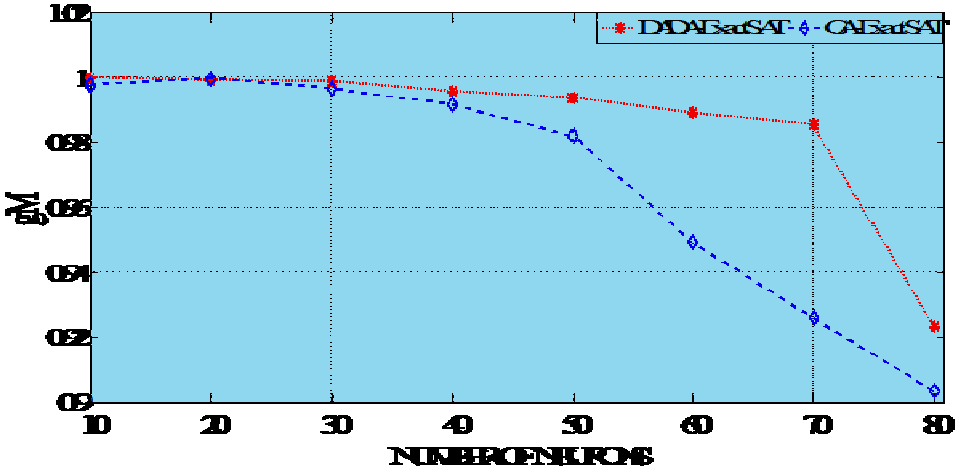

Fig. 7. gM of DADA-Exact*k*SAT and GA- Exact*k*SAT

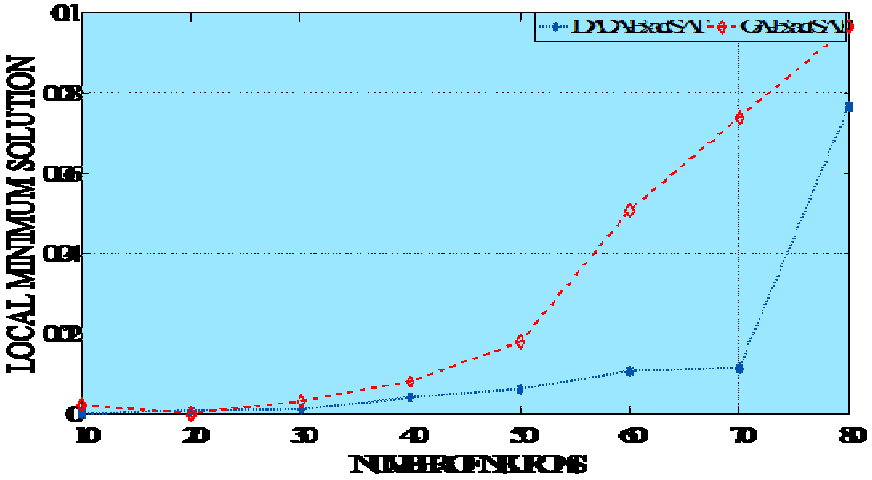

Fig. 8. *Lm* of DADA-Exact*k*SAT and GA- Exact*k*SAT

Figs. 7 and 8 display the apparent success of the DADA in comparison with GA in generating the optimally satisfied clauses on ABM Netlogo. According to Figs. 7 and 8, Gm and Lm of both DADA-Exact *k*SAT and GA-Exact *k*SAT from $10 \le NN \le 100$. The ability of metaheuristics algorithm like DADA and GA to control a high number of neurons from $10 \le NN \le 100$ may be due to the sheer potential of their control parameters mechanism. In DADA and GA model some neurons states get trapped at local solution space as displayed in Fig. 5. In DADA neural get stuck when $NN \ge 80$, affecting 30% while at $NN = 50, 60, 100$, affecting 30% of *NN* generated in GA. DADA model indicates a greater efficiency in neuro-searching in the optimal representation of Exact *k*SAT logic by showing good agreement with the existing model displaying closed to 99% success. As the constraints expand forever, the network becomes more difficult as the *NN* rises in terms of the program's complexity. The clarification on the connection between the global minimum ratio and the existence of the energy obtained at the end of the computation cycle has been elaborated in [40-42]. Potentially, if the global minimum ratio of the implemented hybrid network is close to one, almost all searching has achieved a global minimum solution.

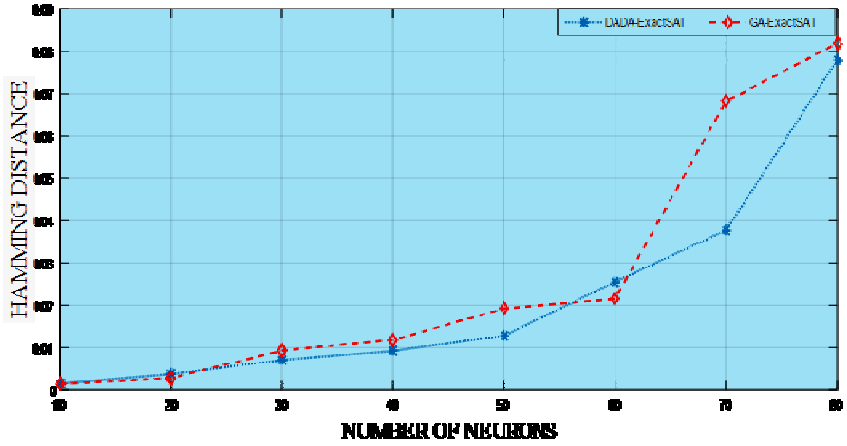

Fig. 9. HD of DADA-Exact*k*SAT and GA- Exact *k*SAT

Fig. 9, displays the trend for HD between DADA and GA based Exact SAT logic representation. It recorded the models' performance from $10 \leq NN \leq 100$. It is explicitly shown that DADA reported agreeing with the performance reported by GA based on HD. They both displays a similar rise in errors as the number of neurons increases from $50 \leq NN \leq 100$ and manifests close margin from $10 \leq NN \leq 40$. In terms of HD assessment, DADA is an acceptable approach doing Exact*k*SAT logic program. It can be observed that DADA is proven to good agreement with GA in term of HD which is much closer to zero in both DADA and GA despite the increasing complexity by increasing the numbers of neurons (NN). This is because both help neurons to move efficiently from global states to stable states. The DADA feature help neurons to reach the energy relaxation looping hence the gap (distance) between the global solutions and stable states are very close to zero.

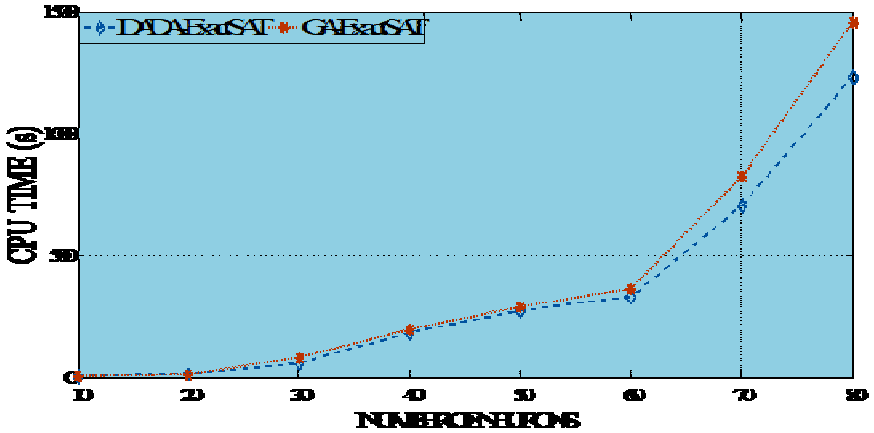

Fig. 10. CPU time of DADA-exact *k*SAT and GA-Exact*k* SAT

Fig. 10 demonstrates the Execution time for our proposed model, DADA in comparison to GA models. A glance at the running time shows that the program is becoming more complex, that it takes more effort to find global solutions. According to Fig. 8, our proposed model DADA-Exact *k*SAT requires more execution time compared to GA-Exact*k*SAT. Due to the fact that more many control parameters are involved in DADA which are required to cross the energy barrier to relax in global searching [43].

9 Discussion

This section presents the performance analysis of the proposed DADA when compared with GA in terms of the Exact*k*SAT logic programs. The chances of getting more clauses satisfied (optimal) improved significantly as artificial dragonfly reinforcing their diversification mechanism and qualities through collision avoidance strategy. The collision avoidance strategy mechanism hopefully attracts an unsatisfied clause into a satisfied clause, this will force the model to converge to optional solution space quickly. Sometimes Exact*k*SAT clause in the solution space may fail to give the best solution, revealing a defect that the algorithm tends to be trapped in local optima. To avoid the problem of premature convergence, balance the efficiency and global search ability of the model. DADA applies the velocity matching mechanism to explore other areas of the solution space to keep the clause away from an unsatisfied solution before exploring the entire solution space. This effort leads up to find the "new" search space which helps the network to avoid local maxima (non-improving solution). Therefore, because the flock centering structure of the DADA model ability to combat uncertainty, the probability for global solutions is increased. As a result, these mechanisms will permit the network to search and compute the feasible solutions within a reasonable execution time. This explores higher feasibility of neuro-searching capacity exhibits by the DADA model.

The searching techniques will work extensively throughout the searching for Exact *k*SAT optimal representation as the complexity of the neurons rises. The justification for this trend is that a higher number of neurons convoluted. DADA problem is to find the consistent Exact *k*SAT mapping since the chances of finding a clear solution significantly reduced. The DADA is more successful in enhancing or pursuing desired approaches, even in the case of literals of high complexity. This is attributed to the various control parameter involved in the mechanism. GA is confirmed to have completed the learning process slightly quicker than DADA, this is due to many operators involves in DADA that delays the searching processing. However, all DADA remain competent in optimizing the Exact *k*SAT and its variant and compute the global solution within a feasible CPU timeframe

10 Conclusion

In this paper, DADA has been developed in agent-based modelling to carry out Exact *k*SAT logic programming by using NETLOGO as a platform. The agent-based modelling that was developed was user friendly. The benefits of ABM over other modelling techniques for DADA-Exact *k*SAT can be captures due to emergent phenomena to produce a model for the set of logic programs. The Ability for ABM in providing a natural description of the framework that incorporates DADA for Exact *k*SAT logic program and for its flexibility to change the training parameters according to the user requirement. The solution monitors and plot show how the algorithm is performing and the best solution that has been found**.**

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Rushdi AMA, Ahmad W. Finding All Solutions of the Boolean Satisfiability Problem, If Any, via Boolean-Equation Solving. Journal of King Abdulaziz University: Engineering Sciences. 2016;27(1).

[2] Karaboga D, Gorkemli B. Solving travelling salesman problem by using combinatorial artificial bee colony algorithms. International Journal on Artificial Intelligence Tools. 2019;28(1):1950004.

[3] Bartz-Beielstein T, Zaefferer M. Model-based methods for continuous and discrete global optimization. Applied Soft Computing. 2017;55:154-167.

[4] Garey MR, Johnson DS. Computers and intractability. San Francisco: Freeman. 1979;174

[5] Carastan-Santos D, de Camargo RY, Martins Jr DC, Song SW, Rozante LC. Finding exact hitting set solutions for systems biology applications using heterogeneous GPU clusters. Future Generation Computer Systems. 2017;67:418-429.

[6] Shi L, Cai X. An exact fast algorithm for minimum hitting set. In 2010 Third International Joint Conference on Computational Science and Optimization. IEEE. 2010;1:64-67

[7] Bisoyi SK, Reza H. Toward securing cyber-physical systems using exact cover set. J Inform Tech Softw Eng. 2017;7(198):2.

[8] Fomin FV, Gaspers S, Lokshtanov D, Saurabh S. Exact algorithms via monotone local search. Journal of the ACM (JACM). 2019;66(2):1-23.

[9] Korula N, Pál M. Algorithms for secretary problems on graphs and hypergraphs. In International Colloquium on Automata, Languages, and Programming. Springer, Berlin, Heidelberg. 2009;508-520.

[10] Guo J, Moalic L, Martin JN, Caminada A. Exact graph coloring algorithms of getting partial and all best solutions. In ISAIM; 2018.

[11] Xiao M, Nagamochi H. Exact algorithms for maximum independent set. Information and Computation. 2017;255:126-146.

[12] Dantsin E, Goerdt A, Hirsch EA, Kannan R, Kleinberg J, Papadimitriou C, Schöning U. A deterministic (2− 2/(k+ 1)) n algorithm for k-SAT based on local search. Theoretical Computer Science. 2002;289(1): 69-83.

[13] Xu X, Yuan H, Liptrott M, Trovati M. Two phase heuristic algorithm for the multiple-travelling salesman problem. Soft Computing. 2018;22(19):6567-6581.

[14] Badrloo S, Kashan AH. Combinatorial optimization of permutation-based quadratic assignment problem using optics inspired optimization. J. Appl. Res. Ind. Eng. 2019;6(4):314-332.

[15] Lunardi WT, Voos H. Comparative study of genetic and discrete firefly algorithm for combinatorial optimization. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing. 2018;300-308.

[16] El Idrissi AL, Tajani C, Krkri I, Fakhouri H. Immune based genetic algorithm to solve a Combinatorial optimization problem: Application to traveling salesman problem. In International Conference on Advanced Intelligent Systems for Sustainable Development. Springer, Cham. 2018;906-915.

[17] Blum C, Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR). 2003;35(3):268-308.

[18] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing. 2011;11(6):4135-4151.

[19] Juan AA, Faulin J, Grasman SE, Rabe M, Figueira G. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. Operations Research Perspectives. 2015;2:62-72.

[20] Essaid M, Idoumghar L, Lepagnot J, Brévilliers M. GPU parallelization strategies for metaheuristics: A survey. International J. of Parallel, Emergent and Distributed Systems. 2019;34(5):497-522.

[21] Salhi S, Eglese R, Hadjiconstantinou E. Preface: Advances in theoretical and practical combinatorial optimization. Annals of Operations Research. 2019;272(1-2):1-2.

[22] Pei XB, Yu XY, Wang SL. Solution of traveling salesman problem by hybrid imperialist competitive algorithm. Journal of ZheJiang University (Engineering Science). 2019;53(10):2003-2012.

[23] Tessaro Lunardi W, Voos H, Cherri LH. An effective hybrid imperialist competitive algorithm and tabu search for an extended flexible job shop scheduling problem. In 34th ACM/SIGAPP Symposium On Applied Computing, Limassol, Cyprus; 2019.

[24] Liu Y, Cao B, Li H. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. JSP. 2020;24(25):54.

[25] Bandaru S, Deb K. Metaheuristic techniques. Decision sciences: Theory and Practice. 2016;693-750.

[26] Mirjalili S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications. 2016;27(4): 1053-1073.

[27] Sayed GI, Tharwat A, Hassanien AE. Chaotic dragonfly algorithm: An improved metaheuristic algorithm for feature selection. Applied Intelligence. 2019;49(1):188-205.

[28] KS SR, Murugan S. Memory based hybrid dragonfly algorithm for numerical optimization problems. Expert Systems with Applications. 2017;83:63-78.

[29] Suresh V, Sreejith S. Generation dispatch of combined solar thermal systems using dragonfly algorithm. Computing. 2017;99(1):59-80.

[30] Reddy AS, Reddy MD. Optimization of distribution network reconfiguration using dragonfly algorithm. Journal of Electrical Engineering. 2016;16(4):273-282.

[31] Xu L, Jia H, Lang C, Peng X, Sun K. A novel method for multilevel color image segmentation based on dragonfly algorithm and differential evolution. IEEE Access. 2019;7:19502-19538.

[32] Suresh MCV, Belwin EJ. Optimal DG placement for benefit maximization in distribution networks by using Dragonfly algorithm. Renewables: Wind, Water and Solar. 2018;5(1):4.

[33] Crooks AT, Heppenstall AJ. Introduction to agent-based modelling. In Agent-based models of geographical systems. Springer, Dordrecht. 2012;85-105.

[34] Sun Z, Lorscheid I, Millington JD, Lauf S, Magliocca NR, Groeneveld J, Buchmann CM. Simple or complicated agent-based models? A complicated issue. Environmental Modelling & Software. 2016;86: 56-67.

[35] Abar S, Theodoropoulos GK, Lemarinier P, O'Hare GM. Agent Based Modelling and Simulation tools: A review of the state-of-art software. Computer Science Review. 2017;24:13-33.

[36] Sibertin-Blanc C, Therond O, Monteil C, Mazzega P. The entity-process framework for integrated agent-based modeling of social-ecological systems. In *Law, public policies and complex systems: networks in action*. Springer, Cham; 2019.

[37] Dahllöf V, Jonsson P. An algorithm for counting maximum weighted independent sets and its applications. *In Proceedings of the thirteenth annual* ACM-SIAM symposium on Discrete Algorithms. 2000;292-298.

[38] Chen Y, Wang Z. Wavelength selection for NIR spectroscopy based on the binary dragonfly algorithm. Molecules. 2019;24(3):421.

[39] Khunkitti S, Watson RN, Chatthaworn R, Premrudeepreechacharn S, Siritaratiwat A. An Improved DA-PSO Optimization Approach for Unit Commitment Problem. Energies. 2019;12(12):2335.

[40] Sathasivam S. Learning in the recurrent Hopfield network. In 2008 Fifth International Conference on Computer Graphics, Imaging and Visualisation, IEEE. 2008;323-328.

[41] Alzaeemi S, Mansor MA, Kasihmuddin MSM, Sathasivam S, Mamat M. Radial basis function neural network for 2 satisfiability programming. Indonesian Journal of Electrical Engineering and Computer Science. 2020;18(1):459-469.

[42] Hammouri ETA, Samra MA, Al-Betar RM, Khalil Z, Alasmer, Kanan M. A dragonfly algorithm for solving traveling salesman problem," 2018 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia. 2018;136-141.
DOI: 10.1109/ICCSCE.2018.8684963

[43] Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S. Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowledge-Based Systems. 2018;161:185-204.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://www.sdiarticle4.com/review-history/58337