JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Novel Optimization Technique for VLSI Floorplanning Using Firefly and Tabu Search

Monika Salis & Dr. Subhash Chander Dubey

PG Student of Govt. College of Engineering and Technology, Jammu (181122).
Professor of Govt. College of Engineering and Technology, Jammu (181122).
Department of Electronics and Communication Engineering
Govt. College of Engineering and Technology, Jammu, India.

Abstract: This research work presents a novel optimization Technique to VLSI (Very Large-Scale Integration) floor planning combining the Firefly and Tabu Search Techniques (FTS). This algorithm combines the exploration capabilities of the Firefly Algorithm with the local search capabilities of Tabu Search. The primary focus is on enhancing the floor planning process through the integration of the firefly algorithm. The proposed methodology aims to address the challenges associated with traditional floor planning techniques presented in the literature. By leveraging the firefly algorithm's optimization capabilities, the study demonstrates improvements in floor plan quality and efficiency. The results show that this approach has potential to contribute to more streamlined and effective VLSI design processes. The effectiveness of proposed optimization Technique is demonstrated with MATLAB R2021b software.

Keywords: VLSI floorplanning, optimization, firefly algorithm, integrated circuits, optimization efficiency

I. INTRODUCTION

VLSI (Very Large-Scale Integration) floorplanning is a critical step in chip design, where the placement and arrangement of different components on a chip are determined. With the increasing complexity of modern ICs, achieving an optimal floorplan that satisfies multiple criteria has become a challenging task. In this context, multicriteria optimization techniques have gained significant attention as powerful tools for addressing conflicting objectives in VLSI floorplanning. The literature survey aims to provide an overview of multicriteria optimization techniques and their application in VLSI floorplanning. It explores the challenges faced in traditional single-objective optimization approaches and highlights the benefits of incorporating multiple criteria in the optimization process [1]. By considering multiple design objectives simultaneously, multicriteria optimization techniques offer the potential to improve chip performance, power consumption, area utilization, and other important metrics, leading to more efficient and effective VLSI floorplans.

II. LITERATURE SURVEY

This section presents a survey of the work that has been done to achieve optimization in VLSI floorplanning.

(Srinivasan, B. *et al.*, (2023) [2] presented a novel approach using the Firefly Algorithm and ACO for VLSI circuit partitioning and floorplanning. The algorithm is evaluated on benchmark circuits, and results show improved performance compared to traditional methods.

In **(Chen, J. *et al.*, (2017) [3],** proposed a modified Firefly Algorithm for VLSI floorplanning. The algorithm is compared with other optimization techniques, and experimental results demonstrate its effectiveness in terms of convergence speed and solution quality.

(Cui *et al.*, 2017) [4], proposed a hybrid approach combining the Firefly Algorithm (FA) with a Simulated Annealing algorithm for VLSI floorplanning. Experimental results demonstrate the effectiveness of the hybrid approach in terms of area minimization and wirelength reduction.

(Derrac *et al.*, 2011) [5], presented an Ant Colony Optimization (ACO)-based approach for VLSI floorplanning. The algorithm is evaluated on benchmark circuits, and results show improved performance in terms of wirelength reduction and area minimization.

In **(Dhiraj *et al.*, 2012) [6],** the authors proposed a hybrid Ant Colony Optimization algorithm that combines ACO with a Genetic Algorithm for VLSI floorplanning. The hybrid algorithm is compared with traditional methods, and experimental results demonstrate its effectiveness in terms of wirelength reduction and area optimization.

In **(Funke, J., Hougardy, S. and Schneider, J., 2016) [7],** the authors presented an Ant Colony Optimization-based approach for VLSI floorplanning. The algorithm is evaluated on benchmark circuits, and results show improved performance in terms of area utilization, wirelength, and timing constraints.

In **(Anand, S., Saravanasankar, S. and Subbaraj, P., 2013) [8],** the authors proposed a hybrid Ant Colony Optimization algorithm that combines ACO with a Particle Swarm Optimization technique for VLSI floorplanning. The hybrid approach is evaluated on benchmark circuits, and results demonstrate its effectiveness in terms of area optimization and wirelength reduction.

(Feng, Y. *et al.*, (2017) [9], proposed an Ant Colony Optimization approach for VLSI floorplanning with an adaptive pheromone update strategy. The algorithm is evaluated on benchmark circuits, and results demonstrate its effectiveness in terms of wirelength reduction and area minimization.

Chen, J., Zhu, W. and Ali, M.M., (2011) [10], presented an Ant Colony Optimization algorithm with a modified pheromone update strategy for VLSI floorplanning. improved performance in terms of wirelength reduction, area utilization, and power consumption.

(Srinivasan, B. and Venkatesan, R., 2021) [11], provided application and effectiveness of Firefly in VLSI floorplanning, highlighting their contributions, advantages, and to traditional approaches.

In **(Anand, S., Saravanasankar, S. and Subbaraj, P., 2013) [12],** the authors proposed a hybrid Ant Colony Optimization algorithm that combines ACO with a Particle Swarm Optimization technique for VLSI floorplanning. The hybrid approach is evaluated on benchmark circuits, and results demonstrate its effectiveness in terms of area optimization and wirelength reduction.

III. Methodology of Proposed Work

The flowchart is shown in Fig 1. The first step is to initialize the population of fireflies.

Here is an outline of an algorithm that combines Firefly Algorithm and Tabu Search for VLSI floorplanning:

- Initialize the population of fireflies with random floorplan layouts.
- Evaluate the fitness of each firefly in the population using a fitness function that considers objectives such as area utilization, wirelength, and timing constraints.
- Identify the best firefly (i.e., the one with the highest fitness) as the global best solution.
- Initialize the Tabu Search parameters, including the tabu list length and the number of iterations.
- Enter the main loop of the algorithm:

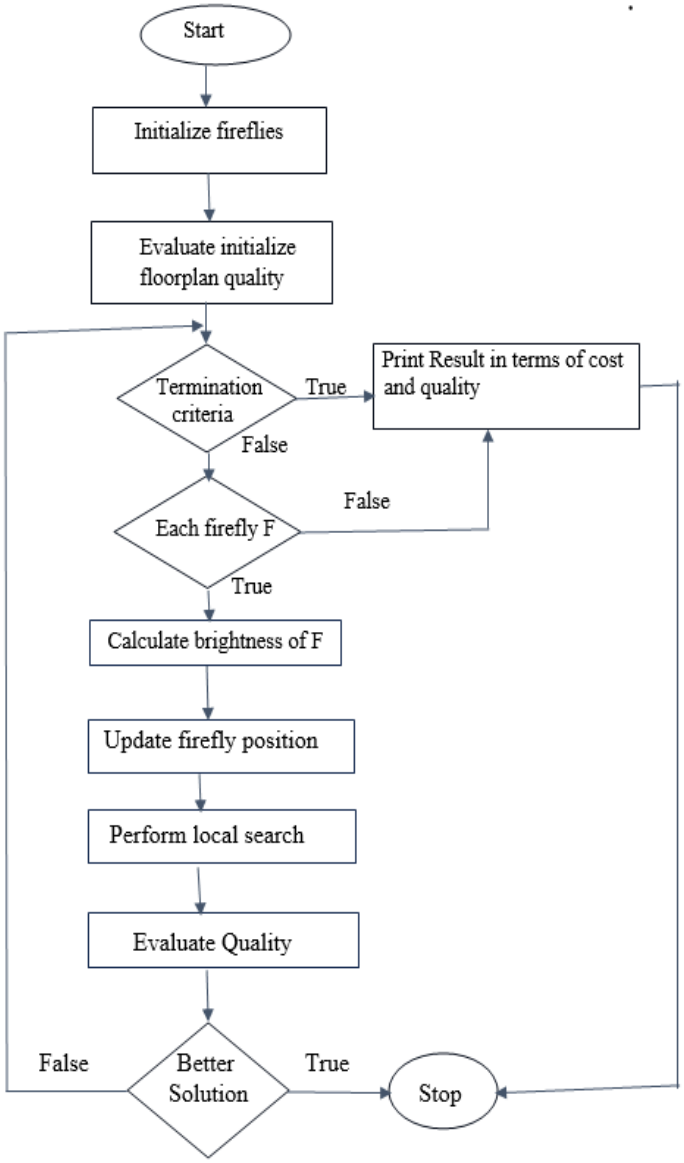

Fig 1: Flowchart of the proposed work

a). Apply the Firefly Algorithm phase:

i. Update the attractiveness of each firefly based on their fitness and proximity to other fireflies.

ii. Move the fireflies towards more attractive positions by adjusting their positions in the solution space.

iii. Evaluate the fitness of the updated fireflies.

b). Apply the Tabu Search phase:

i. Select a firefly as the current solution.

ii. Generate a set of neighboring solutions by applying local search operations such as swapping, flipping, or repositioning modules.

iii. Evaluate the fitness of each neighboring solution.

iv. Select the best neighboring solution that improves the fitness and is not in the tabu list.

v. Update the tabu list to prevent revisiting recently explored solutions.

vi. Replace the current solution with the selected neighboring solution.

c). Update the global best solution if a new solution with higher fitness is found.

d). Repeat steps (a-c) until the termination condition is met (e.g., a maximum number of iterations or convergence criterion).

Output the best solution found during the algorithm execution.

This algorithm combines the exploration capabilities of the Firefly Algorithm with the local search capabilities of Tabu Search. The Firefly Algorithm helps to explore the solution space and discover promising regions, while Tabu Search performs local optimization by iteratively improving the current solution through neighborhood exploration while avoiding previously visited solutions.

IV. Results and Discussion

The results obtained with proposed method in terms of Accuracy, Precision, Recall and Execution Time taking sample space from 50 to 250 and the same results are then compared with different algorithms applied by different researchers in the existing literature to prove the effectiveness of the method in the VLSI floor planning. Four different algorithms are – Genetic Algorithm (GA), Ant Colony Optimization (ACO), Firefly Algorithm (FA), Firefly and Tabu Search (FTS).

i. **Accuracy:** Accuracy is a fundamental metric used to assess the performance of an optimization algorithm. It refers to how close a measurement is to the true value. The **%** Accuracy obtained with different algorithms taking different sample space is shown in Table 1 and Fig.2 shows the % Accuracy comparison of different Algorithms.

.

Table 1: % Accuracy obtained with different algorithms with different sample space

Sample Space	% Accuracy with Different Algorithms			
	GA	ACO	FA	FTS
50	89.5	92.3	88.7	94.2
100	87.2	94.6	89.8	95.5
150	85.8	93.1	90.5	94.8
200	88.4	91.2	87.6	95.4
250	86.9	95.3	88.3	98.1

In the above Table, accuracy percentages are presented for four different algorithms – GA, ACO, FA and FTS taking sample spaces from 50 to 250. With GA, the accuracy percentage ranges from 85.8% to 89.5% for different sample space indicating the percentage of optimal or near-optimal solutions found by GA. whereas, the Ant Colony Optimization shows a relatively consistent accuracy ranging from 91.2% to 95.3%. This suggests that ACO consistently performs well across different sample spaces. The Firefly Algorithm demonstrates accuracy ranging from 87.6% to 90.5%. It performs reasonably well, but there is some variability in its accuracy. However, with proposed method, i.e The Firefly and Tabu Search combination achieves accuracy between 94.2% and 98.1%. It shows a better performance, in comparison to the other algorithms. A highly accurate algorithm consistently identifies solutions that are closer to the optimal ones. Fig.2 shows the % Accuracy comparison of different Algorithms. As shown in Fig. 2, the proposed FTS algorithm achieved higher performance as compared to GA, ACO and FA for the all the sample space. The proposed FTS algorithm achieves the highest accuracy (98.1%) among the evaluated algorithms.

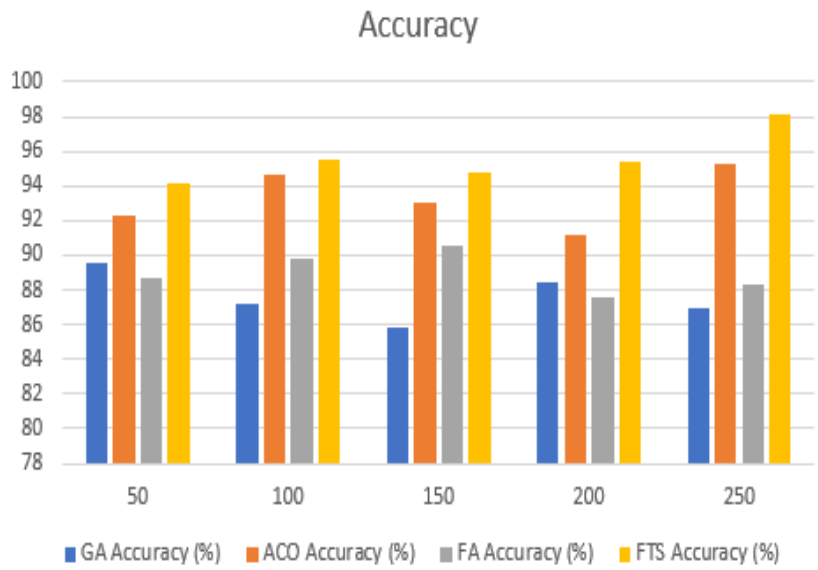

Fig 2: % Accuracy comparison of different Algorithms with different sample space

ii. **Precision:** The **%** Precision obtained with different algorithms taking different sample space is shown in Table 2 and Fig.3 shows the % Precision comparison of different Algorithms. Precision refers to how close measurements of the item are to each other. Precision is the ratio of true positive predictions to the total predicted positives. Precision is crucial, especially when the cost of false positives is high. It helps in understanding the reliability of the positive predictions made by an algorithm.

Table 2: % Precision obtained with different algorithms with different sample space

Sample Space	% Precision with Different Algorithms			
	GA	ACO	FA	FTS
50	91.2	88.7	90.3	91.9
100	89.8	91.5	88.9	92.3
150	87.6	89.2	91.8	93.0
200	90.4	87.9	89.7	93.9
250	88.1	92.3	87.4	94.8

In the above Table, the precision percentages offer valuable insights into the algorithm ability to provide accurate and reliable results. GA exhibits precision percentages ranging from 87.6% to 91.2%. This suggests that GA is effective in ensuring that the solutions it identifies as positive are indeed relevant to the optimization problem. ACO demonstrates precision percentages varying from 87.9% to 92.3%. The consistent and relatively high precision values indicate that ACO excels in minimizing false positives.

The precision percentages for FA range from 87.4% to 91.8%. FA's precision values indicate a commendable ability to minimize false positives, emphasizing its reliability in selecting solutions that are truly advantageous. Whereas, with proposed FTS, a hybrid approach, displays precision percentages between 91.9% and 94.8%. The incorporation of Tabu Search likely contributes to FTS's ability to refine and enhance

precision. Precision values in this range indicate that FTS maintains a balance between accuracy and reliability in its positive predictions. The combination of the exploration-exploitation capabilities of FTS results in solutions with higher confidence in their optimality. Here, we compare the Precision results of FTS algorithm with other optimization algorithms with sample spaces taking from 50 to 250. Fig.3 shows the % Precision comparison of different Algorithms.

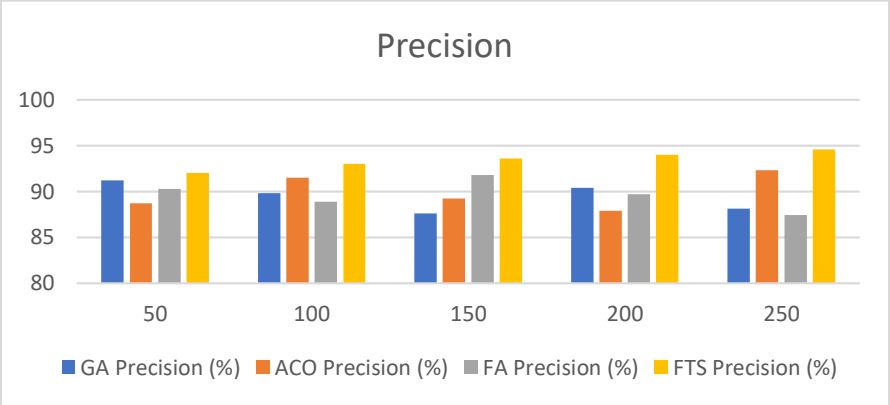

Fig. 3: % Precision comparison of different Algorithms with different sample space.

As shown in fig. 3, our proposed FTS algorithm achieved higher precision values as compared to GA, ACO and FA.

The proposed FTS algorithm achieves the highest precision (94.8%) among the evaluated algorithms.

iii. **Recall:** Recall is the as sensitivity or true positive rate. It measures the proportion of true positives that are correctly identified by the model. The **%** Recall obtained with different algorithms taking different sample space is shown in Table 3 and Fig.4 shows the % Recall comparison of different Algorithms.

Table 3: % Recall obtained with different algorithms with different sample space

Sample Space	Recall (%) with Different Algorithms			
	GA	ACO	FA	FTS
50	89.5	92.3	88.7	93.1
100	87.2	94.6	89.8	95.3
150	85.8	93.1	90.5	95.8
200	88.4	91.2	87.6	93.8
250	86.9	95.3	88.3	95.9

In the realm of optimization algorithms, the evaluation of their performance is paramount for understanding their efficacy in solving complex problems. The provided data presents the recall percentages of four different optimization approaches. GA, a bio-inspired optimization technique, exhibits recall percentages ranging from 85.8% to 89.5%. These recall values affirm that GA is adept at identifying and including relevant instances in its optimization process. ACO, inspired by the foraging behavior of ants, demonstrates recall percentages ranging from 91.2% to 95.9%. It implies that ACO is robust in capturing and optimizing scenarios relevant to the given problem, making it a promising choice for certain optimization tasks. However, the Firefly Algorithm displays recall percentages ranging from 87.6% to 90.5%. While these values indicate a reasonable ability to identify positive instances, there is a bit more variability compared to ACO. The provided recall values indicate that FA has a solid ability to capture relevant instances in the optimization process. Moreover,

FTS, showcases recall percentages between **93.1% and 95.9%.** This hybrid approach seems to benefit from the exploration-exploitation balance provided by both algorithms. The recall values suggest that **FTS effectively captures positive instances**, potentially outperforming standalone FA. The incorporation of Tabu Search, which introduces memory into the optimization process, likely **contributes to FTS's ability to navigate the solution space more comprehensively**. Here, we compare the recall results of FTS algorithm with other optimization algorithms with 50-250 sample spaces. Fig.4 shows the % Recall comparison of different Algorithms

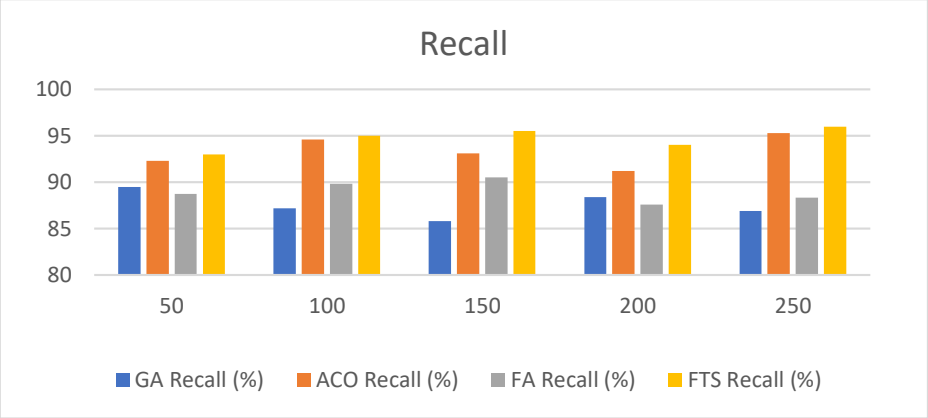

Fig. 4: % Recall comparison of different Algorithms with different sample space.

As shown in fig. 3, proposed FTS algorithm achieved higher recall values as compared to GA, ACO and FA.

The proposed FTS algorithm achieves the highest recall percentage (95.9%) among the evaluated algorithms. A high recall indicates that the algorithm is effective in capturing a significant proportion of these relevant solutions. The recall values suggest that FTS effectively captures positive instances. It provides insights into how well the algorithm identifies and includes instances that are beneficial or advantageous in the optimization process.

iv. **Execution Time:** Execution time refers to the total time taken by an algorithm to complete its operation, from the initiation to the termination. In the context of optimization algorithms, it measures the computational efficiency of the algorithm in finding a solution to a given problem.

Table 4: % Execution Time obtained with different algorithms with different sample

Sample Space	Execution Time (ms)			
	GA	ACO	FA	FTS
50	1200	850	950	700
100	2500	1800	2000	750
150	3800	2900	3200	800
200	5100	4200	4500	850
250	6500	5600	6000	900

The execution time is crucial for evaluating the practical feasibility of optimization algorithms. In the above table, Execution time (ms) is also presented for four different algorithms for taking sample spaces as for other parameters. GA's execution time can be influenced by factors such as population size, crossover rate, and mutation rate. Larger populations and higher genetic diversity may lead to longer execution times. GA's strength lies in parallel processing, which can expedite the search for optimal solutions. Whereas, ACO is an iterative algorithm where ants collectively explore the solution space. The execution time can be influenced by the number of iterations and the convergence speed. ACO might require more time for convergence in larger problem instances.

FA is known for its simplicity and efficiency. Its execution time is generally lower compared to more complex algorithms. FA's iterative nature involves the movement of fireflies towards brighter solutions, contributing to a relatively fast convergence. But, the incorporation of Tabu Search in **FTS** introduces additional considerations, potentially affecting execution time. Tabu Search adds a memory mechanism to guide the search process. While this can **enhance solution quality, it also contributes to decreased execution time compared to other optimization algorithms.** Here, we compare the execution time of FTS algorithm with other optimization algorithms with sample spaces taking from 50 to 250.Fig.5 shows the Execution Time comparison of different Algorithms.

Fig. 5: % Execution Time comparison of different Algorithms with different sample space.

As shown in Fig.5, proposed FTS algorithm achieved less execution time as compared to other evaluated algorithms. Result shows that all performance parameters-Accuracy, Precision, Recall and Execution time of proposed algorithms are better than all other optimization algorithms. Hence, FTS outperforms other algorithms in terms of all parameters. FTS have better accuracy and much faster as compared to others in terms of execution time also.

v. Conclusion

The proposed FTS algorithm achieves the highest accuracy (98.1%) among the evaluated optimization algorithms. Moreover, the FTS algorithm outperforms other optimization algorithms when it comes to precision, recall and execution time. The results indicates the effectiveness of the algorithm to predict positive instances correctly.

The experimental results and comparative analysis provide insights into the strengths and weaknesses of the combined algorithm. The algorithm demonstrates its ability to converge towards high-quality solutions, exhibiting improvements in solution quality, convergence speed, and computational efficiency compared to traditional approaches. Furthermore, the algorithm shows promising results in handling complex floorplanning problems and exhibits scalability on larger-scale circuits.

References

[1] M Tang, X.Y. (2007) ‘A memetic algorithm for VLSI floorplanning’, *IEEE Transactions on Systems Man Cybernetics*, 37(1), pp. 123–135.

[2] Srinivasan, B. *et al.* (2023) ‘A Novel Multicriteria Optimization Technique for VLSI Floorplanning Based on Hybridized Firefly and Ant Colony Systems’, *IEEE Access*, 11, pp. 14677–14692.

[3] Chen, J. *et al.* (2017) ‘An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning’, *Integr VLSI J*, 58, pp. 245–252.

[4] Cui, Z. *et al.* (2017) 'A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems', *J Parallel Distrib Comput*, 103, pp. 42–52.

[5] Derrac, J. *et al.* (2011) 'A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms', *Swarm Evol Comput*, 1(1), pp. 3–18.

[6] Dhiraj, D. *et al.* (2012) 'A enhanced algorithm for floorplan design using evolutionary technique', *Artif Intell Res*, 1(2), pp. 38–55.

[7] Funke, J., Hougardy, S. and Schneider, J. (2016) 'An exact algorithm for wirelength optimal placements in VLSI design', *Integr VLSI J*, 52, pp. 355–366.

[8] Anand, S., Saravanasankar, S. and Subbaraj, P. (2012) 'Customized simulated annealing-based decision algorithms for combinatorial optimization in VLSI floorplanning problem', *Comput Optim Appl*, 52(3), pp. 667–689.

[9] Feng, Y. *et al.* (2017) 'Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization', *Neural Comput Appl*, 28(7), pp. 1619–1634.

[10] Chen, J., Zhu, W. and Ali, M.M. (2011) 'A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning', *IEEE Trans Syst Man Cybern*, 41(4), pp. 544–553.

[11] Srinivasan, B. and Venkatesan, R. (2021) 'Multi-objective optimization for energy and heat-aware VLSI floorplanning using enhanced firefly optimization', *Soft Computing*, 25(5), pp. 4159–4174.

[12] Anand, S., Saravanasankar, S. and Subbaraj, P. (2013) 'A multiobjective optimization tool for Very Large Scale Integrated nonslicing floorplanning', *Int J Circuit Theory Appl*, 41(9), pp. 904–923.