applied sciences

MDPI

*Article*

# Hybrid Nearest-Neighbor Ant Colony Optimization Algorithm for Enhancing Load Balancing Task Management

**Fatma Mbarek and Volodymyr Mosorov ***

Institute of Applied Computer Science—Faculty of Electrical, Electronic, Computer and Control Engineering, Lodz University of Technology, 90-924 Lodz, Poland; mbarek.fatma08@gmail.com
* Correspondence: mosorow@kis.p.lodz.pl

**Abstract:** Many computer problems that arise from real-world circumstances are NP-hard, while, in the worst case, these problems are generally assumed to be intractable. Existing distributed computing systems are commonly used for a range of large-scale complex problems, adding advantages to many areas of research. Dynamic load balancing is feasible in distributed computing systems since it is a significant key to maintaining stability of heterogeneous distributed computing systems (HDCS). The challenge of load balancing is an objective function of optimization with exponential complexity of solutions. The problem of dynamic load balancing raises with the scale of the HDCS and it is hard to tackle effectively. The solution to this unsolvable issue is being explored under a particular algorithm paradigm. A new codification strategy, namely hybrid nearest-neighbor ant colony optimization (ACO-NN), which, based on the metaheuristic ant colony optimization (ACO) and an approximate nearest-neighbor (NN) approaches, has been developed to establish a dynamic load balancing algorithm for distributed systems. Several experiments have been conducted to explore the efficiency of this stochastic iterative load balancing algorithm; it is tested with task and nodes accessibility and proved to be effective with diverse performance metrics.

**Keywords:** ant colony optimization; nearest-neighbor; load balancing; hybrid nearest-neighbor ant colony optimization; distributed computing systems



**Citation:** Mbarek, F.; Mosorov, V. Hybrid Nearest-Neighbor Ant Colony Optimization Algorithm for Enhancing Load Balancing Task Management. *Appl. Sci.* **2021**, *11*, 10807. https://doi.org/10.3390/app112210807

Academic Editor: Federico Divina

Received: 27 September 2021
Accepted: 9 November 2021
Published: 16 November 2021



## 1. Introduction

Distributed computing platforms are becoming increasingly important as cost-effective options to conventional high-performance computing platforms. The distributed systems keep going to expand in size, heterogeneity, and diversity of network resources [1,2]. In such complicated platforms, workload management and load balancing become critical factors in keeping business activities afloat. Workload management has reached the top of business and management engineering research priorities [3,4]. One of the most complex concerns in real-time optimization problems is the robust management of a wide range of workload patterns. Robust management includes a resilient load balancing system [5]. Load balancing is an important element in distributed and parallel environments [6,7], as it is used to achieve maximum use of resources, to avoid node overload, to reduce response time, to avoid network bottlenecks, and to ensure system scalability. The challenge of dynamic load balancing persists as a difficult issue of global optimization because of system structure heterogeneity, requisitions of Quality of Service (QoS) per application and administration of computational resources [8,9].

There are several algorithmic methods for optimizing load balancing problems [10–12]. These optimization algorithms were based on a metaheuristic (or stochastic) nature-inspired paradigm. Most metaheuristic algorithms generate random possible solutions at each iteration to enhance the chances of exploring the whole search space [13]. The first feasible solution can be improved using mechanisms as movement, mutation, exchange, and cooperative perception. The improvement process is then repeated many times until the best

solution is discovered, and the optimization is completed using termination conditions. The stochastic methods use a cost function (or fitness function) [14]. The cost function is a set of agents passing through the solution space. The position of an agents in the solution space represents a set of parameters in the cost function. Thus, the aim of the metaheuristic algorithm is to find the global minimum or global maximum of the cost function and to supply high-quality solutions within a feasible lead time [15].

The swarmintelligence-based (SI-based) algorithms belong to the category of stochastic population-based bioinspired algorithms [16]. SI-based algorithms covered algorithms that are based on the behavior of different animal species, chemical processes, or even other natural processes [17]. Several nature-inspired metaheuristic algorithms have been presented to handle optimization problems in different applications. The ant colony optimization (ACO) [18], genetic algorithm (GA) [19], and particle swarm optimization (PSO) [20] are some examples of algorithms inspired by nature. ACO imitates the behavior of a colony of real foraging ants to find the most cost-effective path. The shortest or optimal path is found via the stigmergy process. It is a social network mechanism where pheromones push agents toward promising solutions. ACO is designed to solve the most challenging concern of combinatorial optimization problems, as well as of network applications, such as routing and load balancing. ACO algorithms can be applied to solve different problems, such as the traveling salesman problem (TSP) [21], vehicle routing [22], sequential ordering [23], and scheduling [24]. GA is a search algorithm based on the concepts of natural selection and natural genetics (crossover and mutation). The GA is employed in real-world optimization problems, for example, manufacturing [25], warehouses [26], robotics [27], and automatic control [28]. PSO mimics the behavior of a swarm of birds or fish in search of food. In PSO, each particle in a swarm does not exchange materials with other particles. A particle is impacted by its current position, the best position in the swarm, and its velocity. Lately, PSO has been used to solve real-world engineering problems such as the design of multilayered rectangular microstrip antenna using electromagnetic band-gap (EBG) structures [29], and bandwidth improvement of an inverted-F antenna (IFA) [30]. Furthermore, the PSO algorithm is also able to design new engineering components, for example, artificial magnetic conductors [31].

Apart from optimizing solutions based on bioinspired algorithms, the multiobjective optimization (MOO) algorithm is designed to solve the most challenging concern of combinatorial optimization problems, as well as of network applications. MOO algorithm is known as Pareto optimization, and it is classified as a multiple-criteria decision analysis (MCDA). The MOO algorithm is used in connection with optimization problems that include more than one conflicting objective function to be optimized simultaneously [32]. Multiobjective problems, such as routing in communication networks [33,34], compressor design [35,36], engineering [37,38], and logistics [39], require a number of objective functions for simultaneous optimization [40].

It is essential to solve the system's task scheduling problem to increase resource utilization and sharing rate in distributed systems. Task scheduling became more difficult and complex for parallel and distributed environment due to resource distribution, heterogeneity, and autonomy. Different research projects have been carried out to improve the ant colony optimization performance, for example, a novel pheromone update strategy [41,42], a modified ant colony optimization with improved tour construction and pheromone updating strategies [43], an AntNet with reward-penalty reinforcement learning [44], and an AntNet routing technique for real-world application [45]. A further improvement on the original form of the ant system (AS) is called the elitist strategy (elitist-AS) [46]. The concept is to provide strong reinforcement at the edges of the optimal path determined at the beginning of the algorithm. Using the elitist method with an appropriate number of elitist ants enables AS to locate better routes earlier in the run. Despite the existing researches for ACO algorithm can improve the efficiency of task scheduling, they do not contribute significantly to improving the overall load imbalance of the system. In this

paper, a novel ACO-NN approach is proposed to manage task assignment of load balancing system. The developed algorithm (ACO-NN) meets the following main contributions:

(i) Take into consideration the system heterogeneity and the dynamic task scheduling;
(ii) Carry out different computing data sets and experiments to investigate load balancing performance;
(iii) Minimize the total cost of the computing system;
(iv) Validate the obtained results with the results of previous research using makespan and the optimal solution as performance measures.

The structure of this paper is organized as follows. Section 2 introduces the hybrid of nearest neighbor and ACO algorithm. In Section 3, various experiments are presented, and the outcomes of the experiments are analyzed. Section 4 carries out the discussion of comparison tests. Finally, conclusion is provided in Section 5.

## 2. Methods

In this research, a new load balancing algorithm based on ant-inspired behavior called hybrid nearest-ant colony optimization (ACO-NN) is proposed. It is properly described in Algorithm 1. ACO-NN contains three strategies: approval procedure, nearest neighbor operator, and ACO operator.

Besides the heuristic and fitness functions of load balancing, the approval procedure $Pre - approve\_Tasks()$ controls the management of workload. ACO-NN can minimize the amount of processing time used in scheduling.

**Algorithm 1** Pseudocode of nearest-neighbor ant colony optimization.

```
Initialization;
Every task comes;
Pre − approve_Tasks();
while any_task_needs_to_be_scheduled do
    Apply NN operator;
    Store the current value of optimal path;
    Update routing table;
    Sort the ants for search using the routing table;
    Path construction;
    Update pheromone table;
    if all ants complete their tour then
        Evaluate updated pheromone table;
        Select the best path;
        if Best_solution_found then
            Choose the optimal node based on pheromone table;
            Assign task to optimal node;
        else
            Find next optimal solution;
        end if
    end if
end while
```

### 2.1. Nearest Neighbor (NN) with Ant Colony Optimization (ACO)

In order to minimize computation time and boost the scheduling result of a total system resource set $N = \{N_1 \dots N_m\}$ and a finite set of tasks $T = \{T_1 \dots T_j\}$, it is important to choose the right job scheduling order through an approval procedure, $Pre - approve\_Tasks()$. The local search phase through the nearest neighbor operator constructs the problem graph and determines the beginning node of ant. The proposed method assumes that the job scheduling mechanism is analogous to the ant foraging process. According to the system resource node's hardware performance parameters and the system average load gap, the pheromone is updated, and the estimated time to execute all

tasks is kept to a minimum. The execution time is the amount of time required by a task to complete its execution. We define $C_i$ to be the execution time of task $i$. The parameters $t_{in}$ and $t_{out}$ represent respectively the arrival time of task $i$ for processing and the complete processing time of task $i$. The general formula for an execution time for a task $i$ is defined by Equation (1):

$$C_i = t_{out} - t_{in} \tag{1}$$

The estimated total execution time of $n$ tasks is represented by Equation (2):

$$C_n = \sum_{i=1}^{n} C_i \tag{2}$$

The heuristic function of the ant colony optimization algorithm is calculated using the estimated completion time. Moreover, to assign a task, a resource node with a strong pheromone concentration (high efficiency and low load), maximum remaining memory, and a short completion time is chosen.

The aim of the proposed task scheduling method, nearest-ant colony optimization (ACO-NN) is to assign each task of the set $T$ to the resource set $N$ on the basis of maintaining load balance and increasing the efficiency of the system's task scheduling. This hybrid algorithm combined ACO with NN to get an efficient and feasible optimization method. Figure 1 displays the flowchart of the specific steps for ACO-NN in distributed systems based on load balancing, while the generic pseudocode of this proposed method is presented in Algorithm 1.

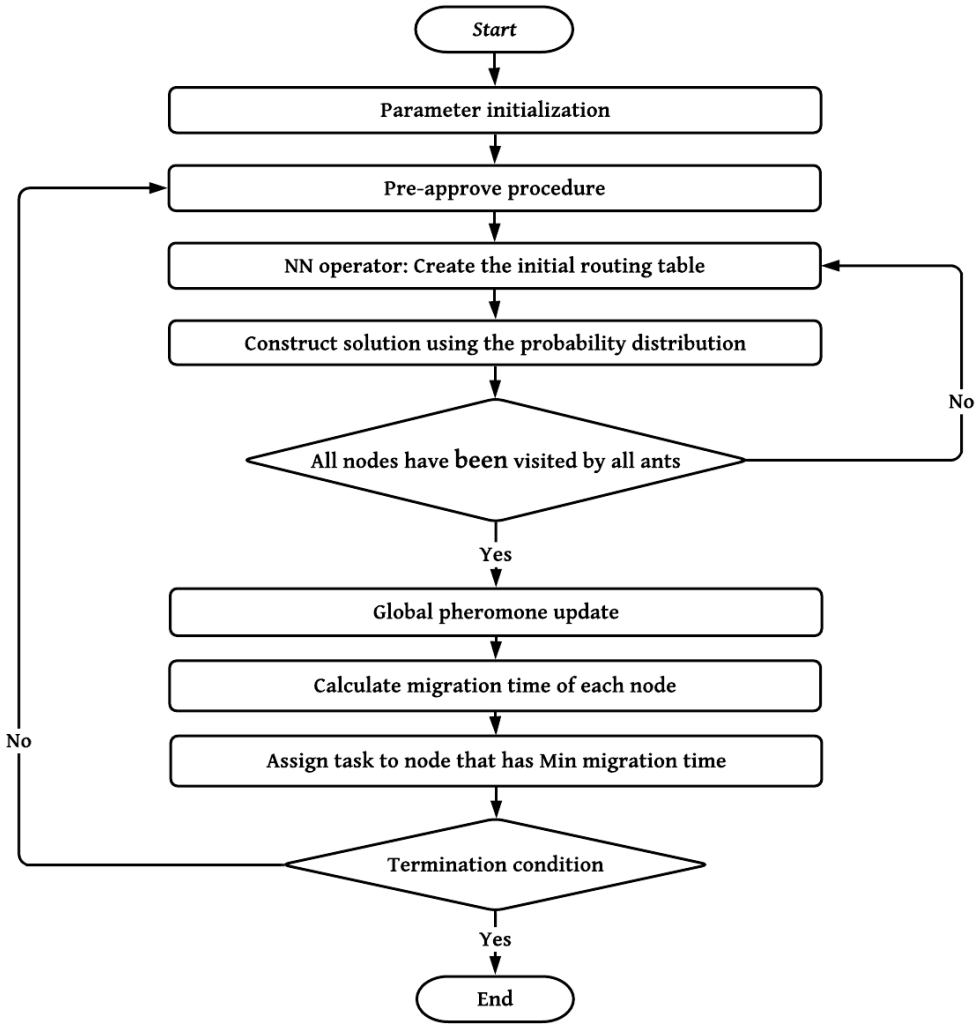


**Figure 1.** The flow chart of nearest-neighbour ant colony optimization task scheduling technique.

### 2.2. Preapprove Procedure

Preapprove is the first stage of ACO-NN at each time tasks arrive. In this stage, the algorithm will verify the available memory of each computing node and determine the maximum amount of memory left over. If the request requires a higher memory than the maximum left memory, the request will be denied prior to scheduling. The preapprove step reduces the size of the ACO solutions, and indeed, the computing time of the ACO scheduling is reduced. Considering $N1$ and $N2$, two service nodes in the system. The residual memory in $N1$ and $N2$ is 2 and 3 GB. Supposing two requests $T1$ and $T2$ are reaching the system with demanding memory of 1 and 5 GB. Whenever a new request comes in, preapprove evaluates the maximum existing memory in each node, which is 3 GB. After this, see if the new request can be approved. Though $T1$ is demanding memory of 1 GB, which is less than the overall remaining memory in a single service node, so $T1$ can be served by any node. While the demanding memory of $T2$ is 5 GB, which is bigger than the total remaining memory of nodes. As consequence, $T2$ will not be approved by $Pre - approve\_Tasks()$ because there is no available resource to serve it. The rejected request will be in queue buffer until one of the available resources can handle it. The request that is in the queue has priority over new incoming requests (in case available resource can serve the pending workload).

### 2.3. Nearest Neighbor Operator

It is a local search strategy used for pattern recognition of the distributed system and construct the routing table for all ants. This is the simplest, easiest, and most straightforward heuristic method to generate the short tour using Euclidean distance calculation.

The Euclidean distance $(D)$ between two nodes $N1$ and $N2$ of $m$ dimensions is obtained by Equation (3):

$$D(N1, N2) = \sqrt{\sum_{i=1}^{m} (N1_i - N2_i)^2} \tag{3}$$

The nearest neighbor is a structured approach that follows the following steps:

step 1 Select a random node.
step 2 Find the nearest unvisited node using a distance calculation.
step 3 If unvisited nodes exist, repeat step 2.

### 2.4. ACO Operator

It is used to build possible solutions for all ants. Each ant will choose the next resource node according to the probability matrix $P_{xy}$ as shown in Equation (4). The value of this transition probability of ant from resource node $x$ to resource node $y$ is used to select the next node to be allocated by a task.

$$P_{xy} = \frac{\tau_{xy}^{\alpha} \times \eta_{xy}^{\beta}}{\sum_{z \in T} (\tau_{xz}^{\alpha} \times \eta_{xz}^{\beta})}, y \in T \tag{4}$$

In Equation (4), $\tau_{xy}$ is the pheromone value for the transition of resource node $x$ to resource node $y$. $\eta_{xy}$ is a heuristic function that represents a priori desirability of the move from resource node $x$ to resource node $y$. $\alpha$ is a positive parameter used to control the influence of pheromone concentrations and heuristic information and which is the relative value of scheduling order; $\beta$ is the expected heuristic component, which sets out the relative heuristic information in the scheduling sequence of selections for an ant. $T$ denotes the set of unscheduled requests that remain. An ant can choose a task among the set $T$ to execute in the next move.

The two main factors influencing the choosing of resource nodes, according to Equation (4), are $\tau_{xy}$ and $\eta_{xy}$. The complexity and emphasis of the research algorithm are considered the improvement of these two main factors. The heuristic function $\eta_{xy}$ is described by Equation (5).

$$\eta_{xy} = \phi_1 \times C(x) + \phi_2 \times M(x) + \phi_3 \times D_u(x) \quad (5)$$

where $\phi_1$, $\phi_2$ and $\phi_3$ are the effect weights of CPU utilization, memory, and disk utilization, respectively; $C(x)$, $M(x)$, and $D_u(x)$ indicate the efficiency of resource node $x$ in the three following resources:

$$C(x) = \frac{Max_{cu} - CU_x}{Max_{cu}} \quad (6)$$

$$M(x) = \frac{RM_x}{Max_m} \quad (7)$$

$$D_u(x) = \frac{Max_{du} - DU_x}{Max_{du}} \quad (8)$$

where $CU_x$ is the CPU utilization for the node $x$ and $RM_x$ and $DU_x$ represent the remaining memory amount and the disk utilization for the node $x$, respectively.

In ACO, the heuristic function corresponds to a local point, which implies that each ant has its decision when it comes to path selection. As a result, each ant chooses its path based on the available resources on each server. In terms of CPU and disk usage, lower utilization means higher resource availability, as shown in (6) and (8). From the standpoint of memory, more remaining memory on the server leads to better efficiency as shown in Equation (7). The sum of these three values represents the average remaining resource as indicated in Equation (5).

### *2.5. Global Pheromone Update Operator*

The proposed ACO-NN method updates pheromone according to the load balancing value and the performance of resource node hardware. Ants lookup the shortest path in the neighborhood according to the current algorithm iteration. Pheromones rise exclusively on the routes that correspond to the current best solution in this iteration, whereas pheromones on the remaining routes decrease with evaporation mechanism. The pheromone trail update is done in accordance with the ant-cycle, where the ants update the pheromone after all the ants have built the tours. The global pheromone update is performed using Equation (9).

$$\tau_{xy}(t+1) = (1-\rho)\tau_{xy}(t) + \sum_{k=1}^{n} \Delta\tau_{xy}^{k}(t) \quad (9)$$

where $1-\rho$ is the global residual coefficient of pheromones decay rate, $0 < \rho \leq 1$. The pheromone trail evaporation prevents bad decisions made before. $\Delta\tau_{xy}^{k}(t)$ represents the amount of pheromones dropped by ant $k$ on arc ($x$,$y$). Generally, $\Delta\tau_{xy}^{k}(t)$ is defined in Equation (10).

$$\Delta\tau_{xy}^{k}(t) = \begin{cases} \frac{Q_k}{L_k}, & \text{if ant } k \text{ passes the arc } (x,y); \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

where $Q_k$ is the quantity of pheromone to be distributed along the route. $L_k$ is the path length performed by ant $k$.

## 3. Results

The proposed approach was developed and implemented in MATLAB environment R2018a. The MATLAB environment was used for numerical computing. In this dissertation, each of the developed algorithms for load balancing were implemented with the aid of MATLAB toolbox, which allows matrix manipulations and the plotting of functions. The algorithms were run with the following computer configuration: Intel Core i5, CPU 2.20 GHz, RAM 12 GB, and Windows 10.

To further explain the efficacy and practicability of the proposed algorithm, we used the problems TSPLIB ([47,48], source by: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html, accessed on 1 June 2021) for experimental studies. All the problems are solved in TSPLIB and the optimal values are given. Table 1 displays experimental data sets.

**Table 1.** Data sets used for testing.

| Instance | Nb.Node | Optimal Solution |
|---|---|---|
| Bayg29 | 29 | 1610 |
| Eil51 | 51 | 426 |
| St70 | 70 | 675 |
| Eil76 | 76 | 538 |
| KroA100 | 100 | 21,282 |
| KroC100 | 100 | 20,749 |
| Eil101 | 101 | 629 |
| Ch150 | 150 | 6528 |
| D198 | 198 | 15,780 |
| Gr120 | 120 | 6942 |
| Pcb442 | 442 | 50,778 |
| Gr666 | 666 | 294,358 |

The optimal solution for each instance of data sets comes from http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html (accessed on 1 June 2021).

ACO-NN is a kind of heuristic algorithm; its performance is influenced by the various parameter values. Table 2 shows the parameters used by the proposed algorithm.

**Table 2.** Parameters settings of test data.

| Parameters | Values |
|---|---|
| $\alpha$ | 0.8–0.9 |
| $\beta$ | 8–35 |
| $\rho$ | 0.4–0.5 |
| Population size | 500–800 |
| Number of nodes | 29–666 |
| Number of tasks | 100–500 |
| Memory demand | 2–10 GB |

The layout of each experiment consists mainly of the total number of resource nodes where the scheduler assigns tasks to the available computing nodes according to system measurement and the optimal tour cost between nodes.

### *3.1. Experiment 1: Comparison ACO-NN to GA and SA Approaches*

Bayg29, Eil76, Gr120, Pcb442, and Gr666 problems are used for testing the performance of ACO-NN. The obtained results are compared with GA and SA.

Figure 2 displays the convergence of a small-scale instance using Bayg29 as an example. ACO-NN performs better in terms of convergence rate while GA has the worst convergence rate.

As can be seen from Figure 3, ACO-NN obtains the optimal solution with the best convergence rate for a small-scale instance using Eil76. SA has the worst performance among the optimization algorithms.

According to Figure 4, ACO-NN has better convergence performance than GA and SA during 10 experimental runs. For medium-scale instance using Gr120, SA has worse performance.

As illustrated in Figures 5 and 6, ACO-NN has minimal tour cost for five data sets with an increase in the number of runs. The performance of ACO-NN and GA is significantly similar for large-scale instances of Pcb442 and Gr666.

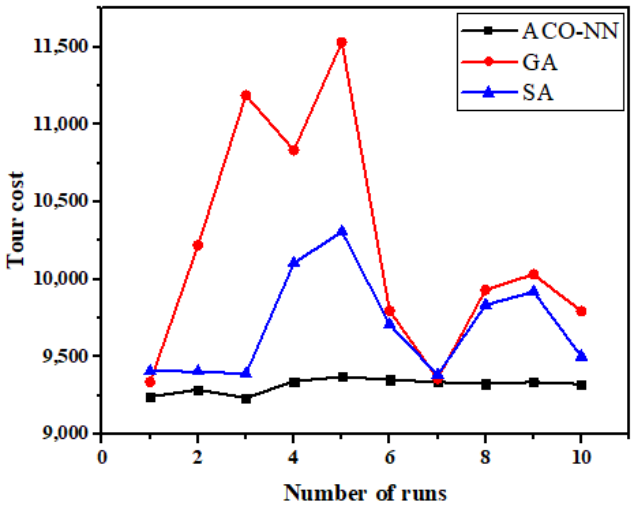


**Figure 2.** Comparison of ACO-NN to GA and SA for Bayg29.

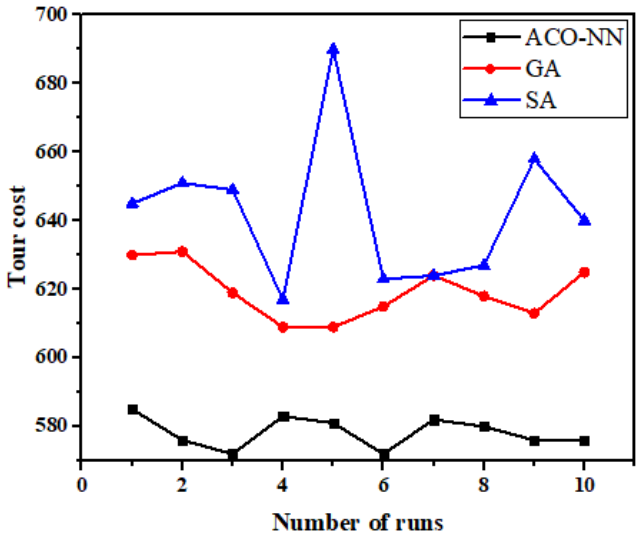


**Figure 3.** Comparison of ACO-NN to GA and SA for Eil76.

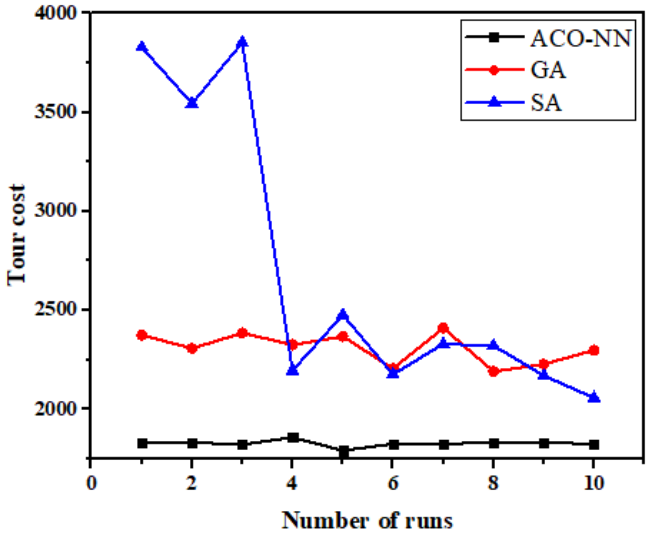


**Figure 4.** Comparison of ACO-NN to GA and SA for Gr120.

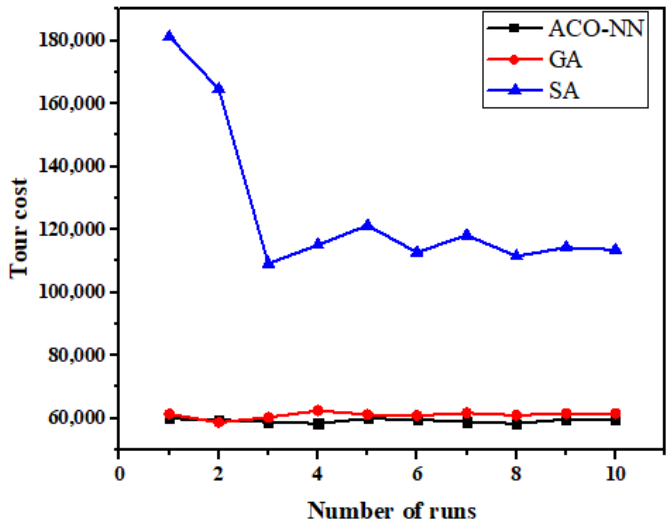


**Figure 5.** Comparison of ACO-NN to GA and SA for Pcb442.

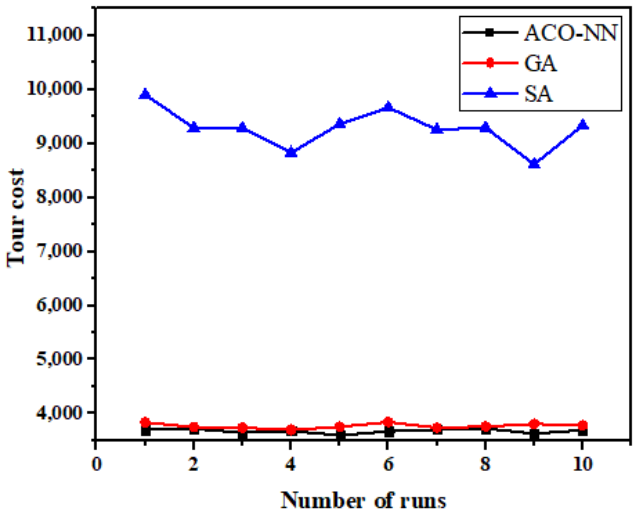


**Figure 6.** Comparison of ACO-NN to GA and SA for Gr666.

### *3.2. Experiment 2: Comparison ACO-NN to GRASP Approach*

To verify the effectiveness of ACO-NN for data sets Eil51, St70 and Kroc100, we compare the proposed algorithm with GRASP.

#### 3.2.1. Results of Optimal Solutions vs. Number of Runs

The comparison of optimal solution based on ACO-NN and GRASP for two data sets with an increase in the number of runs is shown by Figures 7 and 8. The simulation results demonstrate that our proposed algorithm has the lowest value of optimal solutions during 10 runs. ACO-NN is superior to GRASP.

#### 3.2.2. Results of Response Time vs. Number of Runs

The comparison of optimal solution based on ACO-NN and GRASP for two data sets with an increase in the number of runs is shown by Figures 9 and 10. From the perspective of response time, the simulation results shows that our proposed algorithm has minimal response time during 10 runs and it is superior to GRASP.

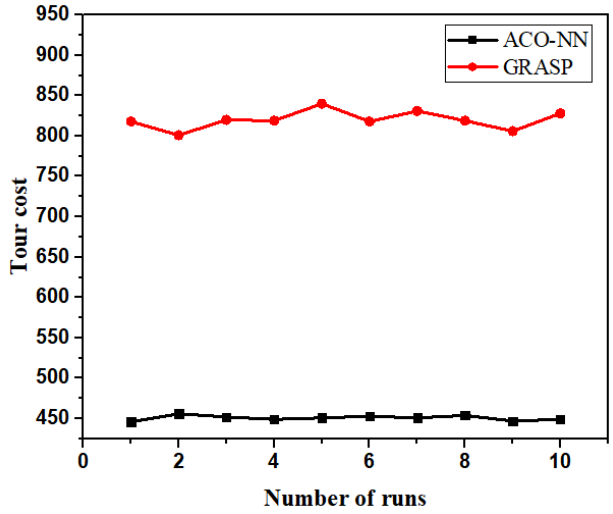


**Figure 7.** Comparison of ACO-NN with GRASP for Eil51 regarding optimal solution metric.

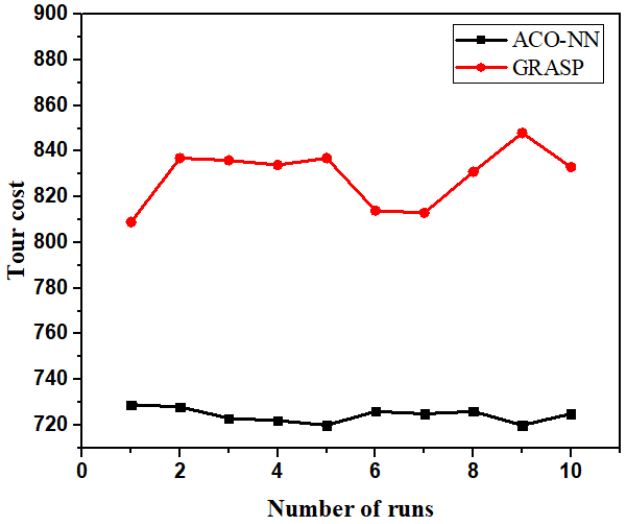


**Figure 8.** Comparison of ACO-NN with GRASP for St70 regarding optimal solution metric.

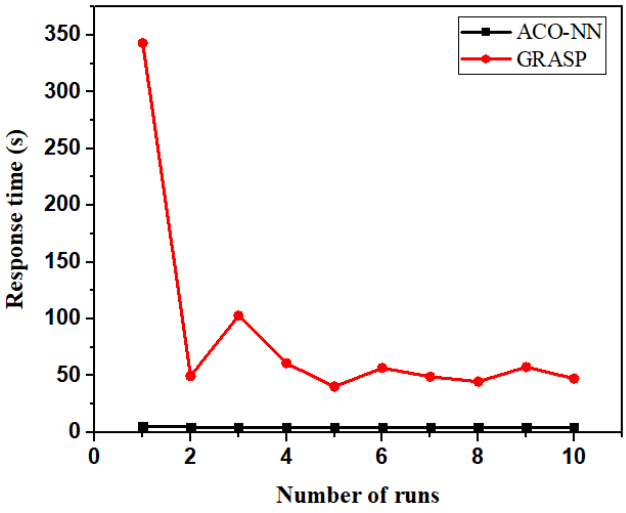


**Figure 9.** Comparison of ACO-NN with GRASP for Eil51 regarding response time metric.

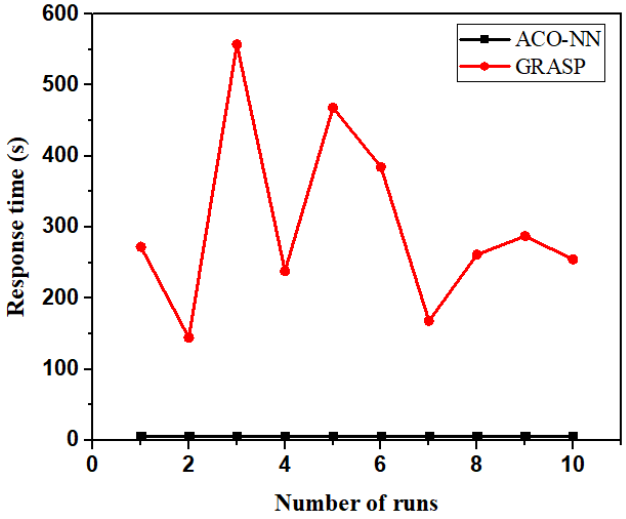


**Figure 10.** Comparison of ACO-NN with GRASP for St70 regarding response time metric.

### *3.3. Experiment 3: Comparison ACO-NN to GA and GRASP Approaches*

The comparison of optimal solution based on ACO-NN, GA, and GRASP in terms of two factors: tour cost and response time, with an increase in the number of runs is carried out by Figures 11 and 12.

According to Figure 11, the simulation results show that our proposed algorithm has the lowest value of optimal solutions during 10 runs for the data set Kroc100 from the perspective of tour cost.

As seen in Figure 12, the results of response time are similar for both algorithms ACO-NN and GA, while the GRASP provides the worst result for the instance KroC100.

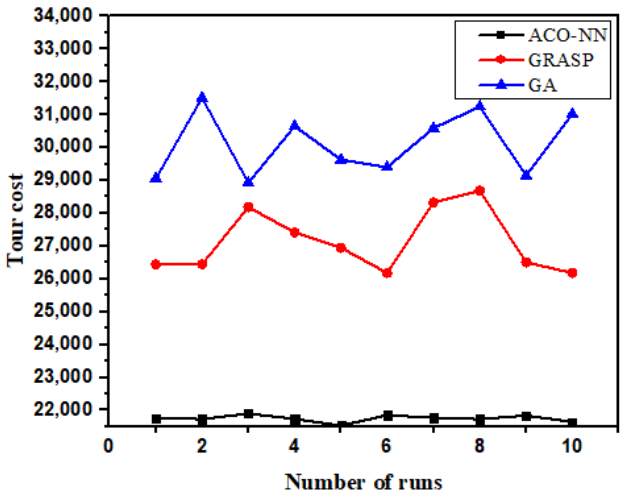


**Figure 11.** Results of optimal solutions vs. number of runs for KroC100.

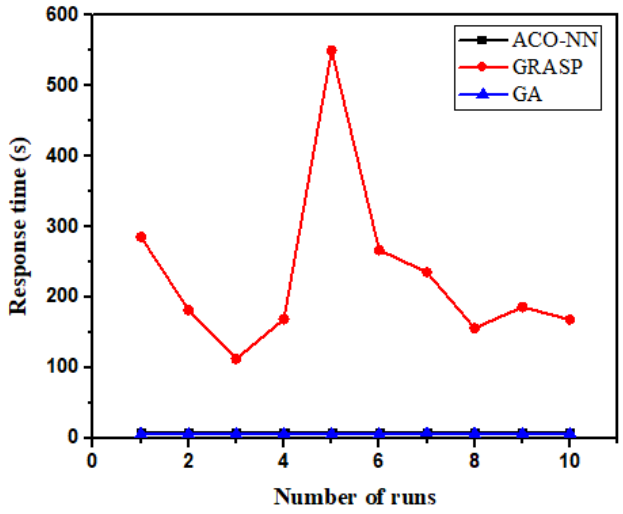


**Figure 12.** Results of response time vs. number of runs for KroC100.

## 4. Discussion

Several tests were carried out to evaluate the performance of the proposed method. The results obtained by the nearest-neighbor ant colony optimization (ACO-NN) algorithm are compared to renowned metaheuristic algorithms such as artificial bee colony (ABC), genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO), camel herd algorithm (CHA), black hole (BH), greedy randomized adaptive search procedure (GRASP), particle swarm optimization (PSO), traveling salesman problem based on simulated annealing and gene expression programming (TSP-SAGEP), simulated-annealing-based symbiotic organisms search (SOS-SA), multioffspring genetic algorithm (MO-GA), and discrete tree-seed algorithm (DTSA) with their variants (DTSA0, DTSAI, DTSAII).

### *4.1. Test1*

In this experiment, three state-of-the-art algorithms based on ACO approaches developed in recent years from the literature were used to test the performance of ACO-NN. The modified ant system was proposed by Yan et al. in 2017 [49]. The adaptive tour construction and pheromone updating techniques are integrated into the standard ant system. The modified ACO (MACO) with improved tour construction and pheromone updating strategies is proposed by Gao in 2021 [43]. The hybrid elitist-ant system (Elitist-AS) with an external memory structure [46] gives strong reinforcement at the arcs of the optimal tour determined at the beginning of the algorithm. The computing results of instance Gr666 are summarized in Table 3.

As demonstrated in Table 3, ACO-NN proves its robustness through the low standard deviation (SD) value and the optimal solution. The comparison results of ACO-NN with Elitist-AS are shown in Table 4. It is noted that ACO-NN is faster than Elitist-AS for instances Eil76, Ch150, and D198. From the perspective of optimal solution value, Elitist-AS performed better than ACO-NN.

**Table 3.** Comparison of ACO-NN to MACO and modified ant system for Gr666.

| Alg. | Nb.Node | Optimal | Best | Mean | Std.Dev | Rank |
|---|---|---|---|---|---|---|
| ACO-NN | 666 | 294,358 | 3597 | 3632 | 83.7198 | 1 |
| MACO [43] | 666 | 294,358 | 294,358 | 294,972.3 | 24,702.1 | 3 |
| Modified ant system [49] | 666 | 294,358 | 294,358 | 294,899.6 | 23,551.35 | 2 |

**Table 4.** Comparison of ACO-NN to Elitist-AS for Eil76, Ch150, and D198.

| Instances | Nb.Node | Optimal | Alg. | Best | Mean | Std.Dev | Time (s) |
|---|---|---|---|---|---|---|---|
| Eil76 | 76 | 538 | ACO-NN<br>Elitist-AS [46] | 562<br>538 | 567<br>551 | 2.8946<br>0 | 5.04<br>42 |
| Ch150 | 150 | 6528 | ACO-NN<br>Elitist-AS [46] | 6846.45<br>6528 | 6914.88<br>6550 | 40.11<br>11.22 | 9.29<br>53 |
| D198 | 198 | 15,780 | ACO-NN<br>Elitist-AS [46] | 17,054.09<br>15,888 | 17,123.04<br>15,940 | 60.83<br>64.83 | 12.48<br>103 |

### *4.2. Test2*

To demonstrate the benefits of the proposed algorithm in this research, we compared ACO-NN to ACO [50], PSO [50], GA [50], BH [50], DTSA [50], and CHA [51] for the instance Eil76 in terms of the meaning of best solutions, the rate of difference (R-mean), and standard deviation. The results are defined in Table 5.

**Table 5.** Comparison of ACO-NN to various approaches for Eil76.

| Alg. | Nb.Node | Optimal | Mean | R-Mean ($\sigma$) | Std.Dev | Rank |
|---|---|---|---|---|---|---|
| ACO [50] | 76 | 538 | 594 | 0.0942 | 40.2152 | 3 |
| PSO [50] | 76 | 538 | 975 | 0.4482 | 152.4061 | 7 |
| GA [50] | 76 | 538 | 652 | 0.1748 | 122.0972 | 4 |
| BH [50] | 76 | 538 | 659 | 0.1836 | 152.1754 | 5 |
| DTSA [50] | 76 | 538 | 588 | 0.0850 | 5.7296 | 2 |
| CHA [51] | 76 | 538 | 687 | 0.2168 | N/R | 6 |
| ACO-NN | 76 | 538 | 567 | 0.0511 | 2.8946 | 1 |

According to Table 5, ACO-NN algorithm is superior in computational results to the other algorithms. In Test 2, the rate of difference was used to assess the merits and disadvantages of experiment outcomes based on ACO-NN and other methods. Equation (11) represents the rate of difference.

$$\sigma = \frac{P_{mean} - P_{optimal}}{P_{mean}} \tag{11}$$

where $\sigma$ defines the rate of difference, the well-known optimal solution is represented by $P_{optimal}$ while $P_{mean}$ describes the mean of the best solution obtained by ACO-NN.

### *4.3. Test3*

In this experiment, we compared ACO-NN to ACO [50], ABC [50], DTSA [50], and CHA [51] for the instance Eil101. As shown in Table 6, the ACO-NN method has good results with the rank 3, and it proves a better experimental result for standard deviation.

**Table 6.** Comparison of ACO-NN to various approaches for Eil101.

| Alg. | Nb.Node | Optimal | Mean | R-Mean ($\sigma$) | Std.Dev | Rank |
|---|---|---|---|---|---|---|
| ACO [50] | 101 | 629 | 693 | 0.0923 | 6.80 | 2 |
| ABC [50] | 101 | 629 | 1315 | 0.5216 | 35.28 | 5 |
| DTSA [50] | 101 | 629 | 689 | 0.0870 | 4.47 | 1 |
| CHA [51] | 101 | 629 | 862 | 0.2703 | N/R | 4 |
| ACO-NN | 101 | 629 | 695 | 0.0949 | 3.70 | 3 |

### *4.4. Test4*

In this test, ACO-NN was compared with other optimization algorithms for the instance KroA100. As illustrated in Table 7, the results revealed that ACO-NN performed much better than the other approaches for optimal solutions and the rate of difference metrics. ACO [50] has a low standard deviation (SD) value. A low SD value indicates that ACO [50] is a reliable algorithm. ACO-NN was in second place for the lower SD results and proves its robustness.

**Table 7.** Comparison of ACO-NN to various approaches for KroA100.

| Alg. | Nb.Node | Optimal | Mean | R-Mean ($\sigma$) | Std.Dev | Rank |
|---|---|---|---|---|---|---|
| ACO [50] | 100 | 21,282 | 22,880 | 0.0698 | 40.2152 | 3 |
| ABC [50] | 100 | 21,282 | 53,840 | 0.6047 | 2198.36 | 6 |
| DSTA0 [50] | 100 | 21,282 | 23,213 | 0.0831 | 906.11 | 4 |
| DSTAI [50] | 100 | 21,282 | 22,835 | 0.0680 | 715.85 | 2 |
| CHA [51] | 100 | 21,282 | 31,786 | 0.3304 | N/R | 5 |
| ACO-NN | 100 | 21,282 | 22,793 | 0.0662 | 162.090 | 1 |

### *4.5. Test5*

The ACO-NN was compared with SA [50], DSTA0, DSTAI, and DTSA [50] for the instance KroC100. According to Table 8, our approach confirmed better results than the other approaches.

**Table 8.** Comparison of ACO-NN to various approaches for KroC100.

| Alg. | Nb.Node | Optimal | Mean | R-Mean ($\sigma$) | Std.Dev | Rank |
|---|---|---|---|---|---|---|
| SA [50] | 100 | 20,749 | 22,223 | 0.0663 | 522.20 | 4 |
| DSTA0 [50] | 100 | 20,749 | 22,877 | 0.0930 | 709.87 | 5 |
| DSTAI [50] | 100 | 20,749 | 21,891 | 0.0521 | 536.88 | 3 |
| DTSA [50] | 100 | 20,749 | 21,817 | 0.0489 | 217.77 | 2 |
| ACO-NN | 100 | 20,749 | 21,753 | 0.0461 | 106.26 | 1 |

As seen from Figure 13, ACO-NN is a greater scheduler in comparison with TSA, DSTA0, DSTAI, and DTSA for the instance Kroc100 from the perspective of rank values.

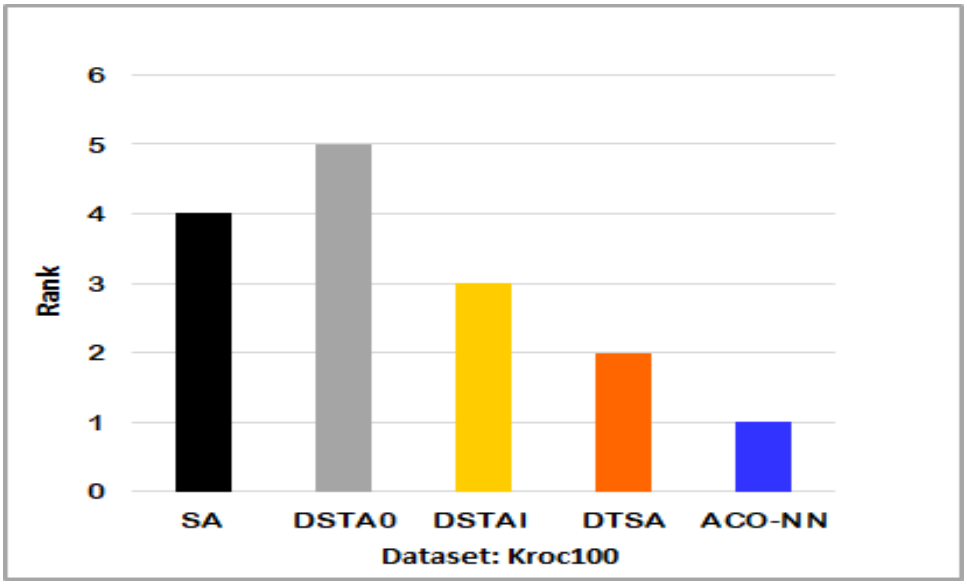


**Figure 13.** Comparison of rank values for ACO-NN, SA, DSTA0, DSTAI, and DTSA.

### *4.6. Test6*

In this experiment, we compare ACO-NN to various approaches such as TSP-SAGEP [52], SOS-SA [53], and MO-GA [54]. For the instances Gr120 and Gr666, our method proves the best results while it provides an acceptable outcome with best execution time for the instance Pcb442. Table 9 displays the following outcomes for different data sets.

**Table 9.** Comparison of ACO-NN to various approaches for Gr120, Pcb442, and Gr666.

| Instances | Nb.Node | Optimal | Alg. | Best | Worst | Average | Time (s) |
|---|---|---|---|---|---|---|---|
| Gr120 | 120 | 6942 | ACO-NN | 1770 | 1843 | 1811 | 6.6174 |
| | | | TSP-SAGEP [52] | 6942 | 7406 | 6995 | 19.4612 |
| | | | SOS-SA [53] | 6942 | 11,237 | 7786 | 25.3581 |
| | | | MO-GA [54] | 6942 | 11,008 | 7655 | 236,709 |
| Pcb442 | 442 | 50,778 | ACO-NN | 56,815 | 58,894 | 58,333 | 40.3890 |
| | | | TSP-SAGEP [52] | 50,811 | 52,147 | 50,878 | 51.0964 |
| | | | SOS-SA [53] | 51,107 | 55,723 | 51,958 | 610,876 |
| | | | MO-GA [54] | 51,097 | 55,006 | 51,828 | 594,571 |
| Gr666 | 666 | 294,358 | ACO-NN | 3597 | 3655 | 3632 | 83.7198 |
| | | | TSP-SAGEP [52] | 294,419 | 305,036 | 295,542 | 79.7853 |
| | | | SOS-SA [53] | 311,855 | 417,702 | 331,024 | 90.0014 |
| | | | MO-GA [54] | 311,003 | 441,298 | 329,199 | 88.0163 |

As seen from Figure 14, ACO-NN is a leader scheduler in comparison with TSP-SAGEP, SOS-SA, and MO-GA for instances GR120 and Pcb442. For the instance Gr666, ACO-NN proves better results than SOS-SA and MO-GA.

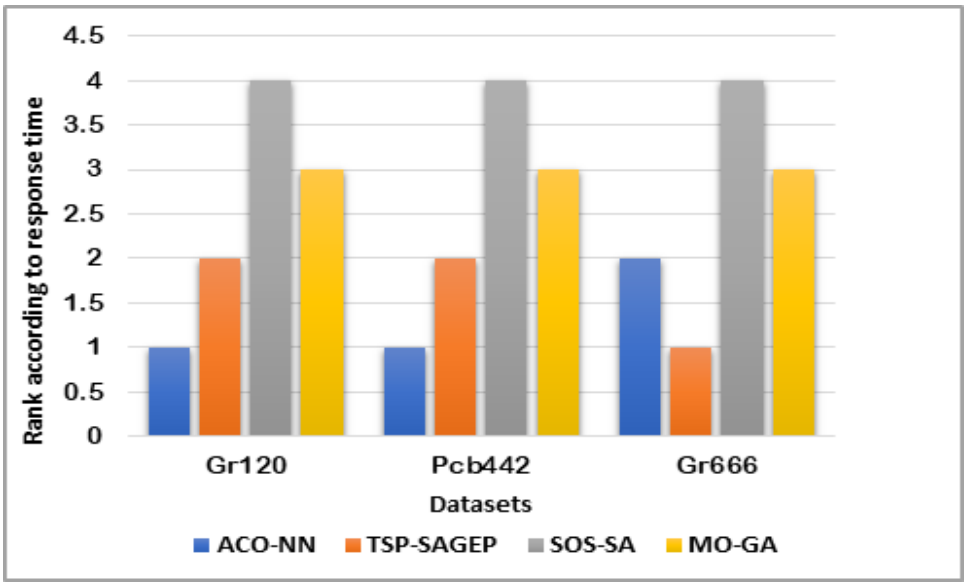


**Figure 14.** Comparison of rank values for ACO-NN, TSP-SAGEP, SOS-SA, and MO-GA.

## 5. Conclusions

This paper addresses load-balancing-related optimization problems impacting the performance of entire systems. The research proposed a ACO-NN approach for load balancing in distributed computing systems to enhance load scheduling mechanism. ACO-NN can manage task scheduling based on node status information. To expedite the ACO process,

we reject dissatisfied requests before scheduling. The preapprove procedure reduces the solution dimensions of the nearest neighbor and ACO, thereby saving computing time in a high-load condition.

To validate the performance of the proposed algorithm, nine trials were carried out on ten benchmark instances taken from the TSPLIB. The experimental results of ACO-NN were compared to fourteen heuristic algorithms. Hybrid ACO-NN proves its efficiency over diverse performance metrics, the experimental results show that the proposed method outperforms existing approaches in maintaining load balance in a dynamic environment.

**Author Contributions:** F.M. and V.M. contributed to the design of research; F.M. implemented and performed the experiments; V.M. and F.M. analysed the study data; V.M. conducted the supervision and validation of the research; and F.M. wrote the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was particularly funded by the Intelligent Development Operational Program 2014–2020 cofinanced by the European Regional Development Fund, project POIR.04.01.04-00-0074/19.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The submitted article contains all of the data, and models developed or used during the study.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ranjan, R.; Rana, O.; Nepal, S.; Yousif, M.; James, P.; Wen, Z.; Barr, S.; Watson, P.; Jayaraman, P.P.; Georgakopoulos, D.; et al. The next grand challenges: Integrating the Internet of Things and data science. *IEEE Cloud Comput.* **2018**, *5*, 12–26. [CrossRef]
2. Rana, B.; Singh, Y.; Singh, P.K. A systematic survey on internet of things: Energy efficiency and interoperability perspective. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4166. [CrossRef]
3. Poola, D.; Salehi, M.A.; Ramamohanarao, K.; Buyya, R. A taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments. In *Software Architecture for Big Data and the Cloud*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 285–320.
4. Thoman, P.; Dichev, K.; Heller, T.; Iakymchuk, R.; Aguilar, X.; Hasanov, K.; Gschwandtner, P.; Lemarinier, P.; Markidis, S.; Jordan, H.; et al. A taxonomy of task-based parallel programming technologies for high-performance computing. *J. Supercomput.* **2018**, *74*, 1422–1434. [CrossRef]
5. Gamoura, S.C. A New Non-Stigmergic-Ant Algorithm to Make Load Balancing Resilient in Big Data Processing for Enterprises. *Artif. Algorithms Natural Algorithms* **2020**, 1–30. Available: https://www.researchgate.net/profile/Samia-Gamoura/publication/353806980_A_New_Non-Stigmergic-Ant_Algorithm_to_Make_Load_Balancing_Resilient_in_Big_Data_Processing_for_Enterprises/links/6112bec4169a1a0103f20303/A-New-Non-Stigmergic-Ant-Algorithm-to-Make-Load-Balancing-Resilient-in-Big-Data-Processing-for-Enterprises.pdf (accessed on 1 June 2021).
6. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load-balancing algorithms in cloud computing: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 50–71. [CrossRef]
7. Semmoud, A.; Hakem, M.; Benmammar, B. A survey of load balancing in distributed systems. *Int. J. High Perform. Comput. Netw.* **2019**, *15*, 233–248. [CrossRef]
8. De Schepper, T.; Latré, S.; Famaey, J. Scalable load balancing and flow management in dynamic heterogeneous wireless networks. *J. Netw. Syst. Manag.* **2020**, *28*, 133–159. [CrossRef]
9. Zeebaree, S.R.; Jacksi, K.; Zebari, R.R. Impact analysis of syn flood ddos attack on haproxy and nlb cluster-based web servers. *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *19*, 510–517.
10. Ebadifard, F.; Babamir, S.M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4368. [CrossRef]
11. Abualigah, L.; Diabat, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust. Comput.* **2021**, *24*, 205–223. [CrossRef]
12. Ding, S.; Chen, C.; Xin, B.; Pardalos, P.M. A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. *Appl. Soft Comput.* **2018**, *63*, 249–267. [CrossRef]
13. Tzanetos, A.; Dounias, G. Nature inspired optimization algorithms or simply variations of metaheuristics? *Artif. Intell. Rev.* **2021**, *54*, 1841–1862. [CrossRef]
14. Nedjah, N.; Mourelle, L.D.M.; Morais, R.G. Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation: A survey-part I. *Int. J. Bio-Inspired Comput.* **2020**, *15*, 207–223. [CrossRef]

15. Agrawal, P.; Abutarboush, H.F.; Ganesh, T.; Mohamed, A.W. Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009–2019). *IEEE Access* **2021**, *9*, 26766–26791. [CrossRef]
16. Slowik, A.; Kwasnicka, H. Nature inspired methods and their industry applications—Swarm intelligence algorithms. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1004–1015. [CrossRef]
17. Fister, I., Jr.; Fister, I. A brief overview of swarm intelligence-based algorithms for numerical association rule mining. *arXiv* **2020**, arXiv:2010.15524.
18. Chen, X.; Yu, L.; Wang, T.; Liu, A.; Wu, X.; Zhang, B.; Lv, Z.; Sun, Z. Artificial intelligence-empowered path selection: A survey of ant colony optimization for static and mobile sensor networks. *IEEE Access* **2020**, *8*, 71497–71511. [CrossRef]
19. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
20. Sengupta, S.; Basak, S.; Peters, R.A. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 157–191. [CrossRef]
21. Yang, K.; You, X.; Liu, S.; Pan, H. A novel ant colony optimization based on game for traveling salesman problem. *Appl. Intell.* **2020**, *50*, 4529–4542. [CrossRef]
22. Huang, Y.H.; Blazquez, C.A.; Huang, S.H.; Paredes-Belmar, G.; Latorre-Nuñez, G. Solving the feeder vehicle routing problem using ant colony optimization. *Comput. Ind. Eng.* **2019**, *127*, 520–535. [CrossRef]
23. Skinderowicz, R. An improved ant colony system for the sequential ordering problem. *Comput. Oper. Res.* **2017**, *86*, 1–17. [CrossRef]
24. Kumar, S.; Solanki, V.K.; Choudhary, S.K.; Selamat, A.; Gonzalez, Crespo, R. Comparative Study on Ant Colony Optimization (ACO) and K-Means Clustering Approaches for Jobs Scheduling and Energy Optimization Model in Internet of Things (IoT). *Int. J. Interact. Multimed. Artif. Intell.* **2020**, *6*, 107–116. [CrossRef]
25. Liu, Z.; Wang, L.; Li, X.; Pang, S. A multi-attribute personalized recommendation method for manufacturing service composition with combining collaborative filtering and genetic algorithm. *J. Manuf. Syst.* **2021**, *58*, 348–364. [CrossRef]
26. Grznár, P.; Krajčovič, M.; Gola, A.; Dulina, L.; Furmannová, B.; Mozol, Š.; Plinta, D.; Burganová, N.; Danilczuk, W.; Svitek, R. The Use of a Genetic Algorithm for Sorting Warehouse Optimisation. *Processes* **2021**, *9*, 1197. [CrossRef]
27. Lamini, C.; Benhlima, S.; Elbekri, A. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Comput. Sci.* **2018**, *127*, 180–189. [CrossRef]
28. Zhang, H.; Zhao, X.; Yang, J.; Zhang, W. Optimizing automatic transmission double-transition shift process based on multi-objective genetic algorithm. *Appl. Sci.* **2020**, *10*, 7794. [CrossRef]
29. Gaharwar, M.; Dhubkarya, D.C. X-Band Multilayer Stacked Microstrip Antenna Using Novel Electromagnetic Band-Gap Structures. *IETE J. Res.* **2021**, 1–10. doi:10.1080/03772063.2021.1883484. [CrossRef]
30. Alnas, J.; Giddings, G.; Jeong, N. Bandwidth improvement of an inverted-F antenna using dynamic hybrid binary particle swarm optimization. *Appl. Sci.* **2021**, *11*, 2559. [CrossRef]
31. Kwon, O.H.; Park, W.B.; Yun, J.; Lim, H.J.; Hwang, K.C. A low-profile HF meandered dipole antenna with a ferrite-loaded artificial magnetic conductor. *Appl. Sci.* **2021**, *11*, 2237. [CrossRef]
32. Malar, A.; Kowsigan, M.; Krishnamoorthy, N.; Karthick, S.; Prabhu, E.; Venkatachalam, K. Multi constraints applied energy efficient routing technique based on ant colony optimization used for disaster resilient location detection in mobile ad-hoc network. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 4007–4017. [CrossRef]
33. Wu, J.; Xu, M.; Liu, F.F.; Huang, M.; Ma, L.; Lu, Z.M. Solar Wireless Sensor Network Routing Algorithm Based on Multi-Objective Particle Swarm Optimization. *J. Inf. Hiding Multim. Signal Process.* **2021**, *12*, 1–11.
34. Vijayalakshmi, K.; Anandan, P. A multi objective Tabu particle swarm optimization for effective cluster head selection in WSN. *Clust. Comput.* **2019**, *22*, 12275–12282. [CrossRef]
35. Sharma, M.; Baloni, B.D.Design optimization of S-shaped compressor transition duct using particle swarm optimization algorithm. *SN Appl. Sci.* **2020**, *2*, 1–17. [CrossRef]
36. Mofid, H.; Jazayeri-Rad, H.; Shahbazian, M.; Fetanat, A. Enhancing the performance of a parallel nitrogen expansion liquefaction process (NELP) using the multi-objective particle swarm optimization (MOPSO) algorithm. *Energy* **2019**, *172*, 286–303. [CrossRef]
37. Lalbakhsh, A.; Afzal, M.U.; Esselle, K.P.; Zeb, B.A. Multi-objective particle swarm optimization for the realization of a low profile bandpass frequency selective surface. In Proceedings of the 2015 International Symposium on Antennas and Propagation (ISAP), Hobart, TAS, Australia, 9–12 November 2015; pp. 1–4. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7447507 (accessed on 1 June 2021).
38. Lalbakhsh, A.; Afzal, M.U.; Esselle, K.P. Multiobjective particle swarm optimization to design a time-delay equalizer metasurface for an electromagnetic band-gap resonator antenna. *IEEE Antennas Wirel. Propag. Lett.* **2016**, *16*, 912–915. [CrossRef]
39. Chan, F.T.; Wang, Z.X.; Goswami, A.; Singhania, A.; Tiwari, M.K. Multi-objective particle swarm optimisation based integrated production inventory routing planning for efficient perishable food logistics operations. *Int. J. Prod. Res.* **2020**, *58*, 5155–5174. [CrossRef]
40. Mbarek, F.; Mosorov, V. Load Balancing Based on Optimization Algorithms: An Overview. *J. Telecommun. Inf. Technol.* **2019**, *4*, 3–12. [CrossRef]

41. Zhao, J.; Li, H.; Yang, C.; Wang, W. A novel path planning method for wheel-legged unmanned vehicles based on improved ant colony algorithm. In Proceedings of the 2021 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Tokyo, Japan, 8–10 September 2021; pp. 696–701.
42. Lalbakhsh, P.; Zaeri, B.; Lalbakhsh, A. An improved model of ant colony optimization using a novel pheromone update strategy. *IEICE Trans. Inf. Syst.* **2013**, *96*, 2309–2318. [CrossRef]
43. Gao, W. Modified ant colony optimization with improved tour construction and pheromone updating strategies for traveling salesman problem. *Soft Comput.* **2021**, *25*, 3263–3289. [CrossRef]
44. Lalbakhsh, P.; Zaeri, B.; Lalbakhsh, A.; Fesharaki, M.N. AntNet with reward-penalty reinforcement learning. In Proceedings of the 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, Liverpool, UK, 28–30 July 2010; pp. 17–21.
45. Wilfert, J.; Paprotta, N.; Kosak, O.; Stieber, S.; Schiendorfer, A.; Reif, W. A Real-Word Realization of the AntNet Routing Algorithm with ActivityBots. Institute for Software and Systems Engineering, University of Augsburg. Available: https://www.researchgate.net/profile/Simon-Stieber/publication/354889888_A_Real-Word_Realization_of_the_AntNet_Routing_Algorithm_with_ActivityBots/links/6152ebed522ef665fb660ab8/A-Real-Word-Realization-of-the-AntNet-Routing-Algorithm-with-ActivityBots.pdf (accessed on 1 June 2021).
46. Jaradat, G.M. Hybrid elitist-ant system for a symmetric traveling salesman problem: Case of Jordan. *Neural Comput. Appl.* **2018**, *29*, 565–578. [CrossRef]
47. Weise, T.; Wu, Z. Difficult features of combinatorial optimization problems and the tunable w-model benchmark problem for simulating them. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, 15–19 July 2018; pp. 1769–1776.
48. Lendl, S.; Ćustić, A.; Punnen, A.P. Combinatorial optimization with interaction costs: Complexity and solvable cases. *Discret. Optim.* **2019**, *33*, 101–117. [CrossRef]
49. Yan, Y.; Sohn, H.S.; Reyes, G. A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem. *Appl. Soft Comput.* **2017**, *60*, 256–267. [CrossRef]
50. Cinar, A.C.; Korkmaz, S.; Kiran, M.S. A discrete tree-seed algorithm for solving symmetric traveling salesman problem. *Eng. Sci. Technol. Int. J.* **2020**, *23*, 879–890. Available: https://www.sciencedirect.com/science/article/pii/S2215098619313527 (accessed on 1 June 2021). [CrossRef]
51. Ahmed, Z.O.; Sadiq, A.T.; Abdullah, H.S. Solving the Traveling Salesman's Problem Using Camels Herd Algorithm. In Proceedings of the 2019 2nd Scientific Conference of Computer Sciences (SCCS), Baghdad, Iraq, 27–28 March 2019; pp. 1–5.
52. Zhou, A.H.; Zhu, L.P.; Hu, B.; Deng, S.; Song, Y.; Qiu, H.; Pan, S. Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming. *Information* **2019**, *10*, 7. [CrossRef]
53. Ezugwu, A.E.S.; Adewumi, A.O.; Frîncu, M.E. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Syst. Appl.* **2017**, *77*, 189–210. [CrossRef]
54. Wang, J.; Ersoy, O.K.; He, M.; Wang, F. Multi-offspring genetic algorithm and its application to the traveling salesman problem. *Appl. Soft Comput.* **2016**, *43*, 415–423. [CrossRef]