GenACO a multi-objective cached data offloading optimization based on genetic algorithm and ant colony optimization

Mulki Indana Zulfa[1,2], Rudy Hartanto[1], Adhistya Erna Permanasari[1], Waleed Ali[3]

[1] Department of Electrical and Information Engineering, Universitas Gadjah Mada, Yogyakarta, Special Region of Yogyakarta, Indonesia
[2] Department of Electrical Engineering, Universitas Jenderal Soedirman, Purwokerto, Central Java, Indonesia
[3] Department of Information Technology, King Abdulaziz University, Kingdom of Saudi Arabia

Corresponding Author:
Rudy Hartanto[1]
Jln. Grafika Kampus No.2, Yogyakarta, 55281, Indonesia
Email address: rudy@ugm.com

Abstract

Background. Data exchange and management have been observed to be improving with the rapid growth of 5G technology, edge computing, and the Internet of Things (IoT). Moreover, edge computing is expected to quickly serve extensive and massive data requests despite its limited storage capacity. Such a situation needs data caching and offloading capabilities for proper distribution to users. These capabilities also need to be optimized due to the experience constraints, such as data priority determination, limited storage, and execution time.

Methods. We proposed a novel framework called Genetic and Ant Colony Optimization (GenACO) to improve the performance of the cached data optimization implemented in previous research by providing a more optimum objective function value. GenACO improved the solution selection probability mechanism to ensure a more reliable balancing of the exploration and exploitation process involved in finding solutions. Moreover, the GenACO has two modes: cyclic and non-cyclic, confirmed to have the ability to increase the optimal cached data solution, improve average solution quality, and reduce the total time consumption from the previous research results.

Result. The experimental results demonstrated that the proposed GenACO outperformed the previous work by minimizing the objective function of cached data optimization from 0.4374 to 0.4350 and reducing the time consumption by up to 47%.

Introduction

The conduct of work and school activities from home is the new habit adopted during the Covid-19 pandemic Nimrod (2020). It led to a massive surge in internet and digital technology users (De', Pandey, and Pal 2020). Government and business owners are also required to maximize websites

41 and social media to disseminate information Haman (2020), while an increase was also
42 experienced in the use of e-learning and e-banking platforms (Azlan et al, 2020; Naeem and
43 Ozuem, 2021). Meanwhile, data exchange and management have been observed to be improving
44 with the rapid growth of 5G technology, edge computing, and the Internet of Things (IoT) (Sai,
45 Fan, and Fan, 2020; Zhang, 2020). Moreover, edge computing is expected to quickly serve
46 extensive and massive data requests despite its limited storage capacity Wang et al. (2019). Such
47 a situation needs data caching and offloading capabilities for proper distribution to users Shi et al.
48 (2016). These capabilities also need to be optimized due to several constraints the experience, such
49 as data priority determination and limited storage and execution time Bala and Chishti (2019).

50 The data offloading mechanism is also used in a web caching strategy known as cache
51 replacement (Ali, Shamsuddin, and Ismail, 2011) with the focus on cached data management
52 strategies to ensure faster data requests through cache servers (Ali, Mariyam, and Samad , 2012a;
53 Ali, Shamsuddin, and Ismail, 2012b). Moreover, direct data requests on the origin server cloud are
54 required to be suppressed as possible to reduce the communication latency on the user side and
55 bandwidth efficiency on the server side (Mertz and Nunes, 2017; Mertz and Nunes, 2018).
56 However, the data offloading process in this cache replacement also has the same main problem:
57 determining cached data priority due to the limited cache server capacity Tatar et al. (2014).

58 The development of 5G networks, IoT, and intelligent devices requires a reliable edge network
59 for data distribution and interconnection to maintain Quality of Service (QoS) (Prerna,
60 Tekchandani, and Kumar, 2020; Carvalho et al, 2020). A reliable data offloading on the edge
61 network can save 30%-40% energy and maximize caching system performance (Shi et al, 2016;
62 Ha et al, 2014). Figure 1 illustrates the data offloading process in a fog computing environment
63 with data priority and server capacity issues. Some of the data is not included in the edge network
64 caching system due to limited storage capacity. Therefore, the edge network should assess the
65 priority of cached data to minimize data service requests made to the origin server on the cloud
66 network. For example, which data should be stored in the caching system (Z) when the storage
67 capacity on the network edge is 1200 MB? Z={1,2,3} or Z={1,2,4} or Z={3,4,5} or Z={2,5} or a
68 combination of other data. It is important to note that the selected cached data should have the
69 maximum profit.

70 In previous studies, several methods such as Long Short Term Memory (LSTM) (Pescosolido,
71 Conti, and Passarella 2019), Reinforcement Learning (RL) Huang et al. (2019), and Branch and
72 Bound (BnB) Elgendy et al. (2019) have been used to solve problems in optimizing data
73 offloading. However, these methods still leave issues such as being easily trapped at the local
74 optimum, high time complexity, and not focusing on the priority of cached data. The utilization of
75 the server cache is not optimal. Such a situation can cause the edge server to request the same data
76 repeatedly to the origin server. Therefore Wang et al. (2019) conducted a study on cached data
77 offloading optimization based on Swarm Intelligence (SI) using the ACO-GA hybrid algorithm.
78 The SI method effectively solves optimization problems because of its reliability in finding the
79 (near) optimum solution Ertenlice and Kalayci (2018).

80   The Cyclic ACO-GA proposed by Wang et al. (2019) can solve the cached data offloading
81   problem. However, the dominant iteration issued to ACO makes the execution time quite long. In
82   addition, the mechanism for selecting a solution using a roulette wheel makes the resulting solution
83   does not converge quickly. Moreover, the iteration restrictions on the GA algorithm make GA
84   performance not optimal. Hence, the main research question is "how can we hybrid and enhance
85   the performance of evolutionary algorithms to optimize the cached data offloading ?"

86   Therefore in this study, we propose a new GenACO framework to improve Cyclic Genetic Ant
87   Colony (CGACA). This paper proposed a hybrid method of the ACO-GA algorithm to solve the
88   cached data offloading problem. The contributions are (i) improving solution probability
89   calculation on cached data offloading (ii) proposed a novel framework named GenACO with cyclic
90   and non-cyclic modes to improve previous research using more optimal profit cached data.

91   The following section discusses the previous related works on cached data offloading. The
92   algorithms section comprehensively describes the working principle of ACO, GA, and CGACA
93   framework algorithms and their performance in this research. Moreover, the methodology section
94   discusses the proposed performance improvement of the ACO-GA hybrid as tested on a novel
95   GenACO framework with cyclic and non-cyclic modes, followed by the dataset, simulation setup,
96   results, and discussion sections and concluded.

97

98   **Related Works**

99   Cached data offloading is an extraordinary capability required to be owned by an application,
100   workstation, server, and network to manage data storage considered to be larger than its capacity
101   (Zulfa, Hartanto, and Permanasari 2020). It is a very common method used in cloud computing
102   (Wang et al, 2019; C. Li et al, 2020), mobile computing (Dutta and Vandermeer, 2017; Zhu and
103   Reddi, 2017), operating system (Abraham Silberschatz, Peter B. Galvin, 2008; Tian and Liebelt,
104   2014), and telecommunication Prerna et al. (2020).

105   Luo et al. (2017) examined the energy consumption optimization in Mobile Edge Computing
106   (MEC) by formulating objective functions based on the variables of energy consumption, backhaul
107   capacities, and content popularity. The research utilizes the GA algorithm with the optimization
108   function to minimize the energy consumption value and validated by measuring the system average
109   delay and average power toward adding the MEC servers and backhaul capacity. Moreover, Xu et
110   al. (2019) also proposed a data offloading optimization framework named COM, designed to
111   optimize mobile devices execution time and energy consumption. The COM framework was used
112   to model a multi-objective optimization solved using the NSGA-III algorithm. It was validated by
113   calculating the maximum value of the utility and resource utilization functions achieved.

114   Pescosolido et al. (2019) developed a Content Delivery Management System (CDMS) method
115   for device-to-device data offloading in cellular networks. It was proposed to predict system
116   performance in transmission distance and energy consumption using simulations configured with
117   Infrastructure-to-Device (I2D) transmission and Base Stations. The research was validated through
118   the efficiency of data offloading and Physical Resource Block (PRB). Furthermore, Zhao et al.
119   (2019) proposed mobile data offloading by predicting the real-time traffic Small Base Station

120   (SBS) on MEC using a multi-Long Short-Term Memory (LSTM) algorithm. The results predicted
121   were further used for data offloading by utilizing the Cross-Entropy (CE) algorithm. The research
122   formulated the problem with an optimization approach to determine the maximum value from the
123   system throughput.
124   Huang et al. (2019) also developed the Deep-Q Network (DQN) framework to solve data
125   offloading optimization and resource allocation in MEC using energy costs, computation costs,
126   and delay costs as variables. The optimization was modeled with mixed-integer non-linear
127   programming solved by Reinforcement Learning (RL) algorithm, and the DQN was validated by
128   calculating the minimum total cost generated. Another research by Elgendy et al. (2019) focused
129   on resource allocation optimization and computation offloading with additional data security
130   protection in MEC. The security protection was conducted by adding the Advanced Encryption
131   Standard (AES) algorithm to prevent cyber-attacks. At the same time, the optimization problem
132   was modeled based on the knapsack problem and solved using a branch and bound algorithm. Its
133   implementation was further validated by determining the smallest value of time and energy
134   consumption.
135   Kuang et al. (2021) modeled data offloading and resource allocation in one cooperative scheme
136   to optimize power allocation and CPU cycle frequency in MEC and, subsequently, to make
137   appropriate data offloading decisions. The research was measured by calculating the smallest
138   possible task latency value and also adopted the Convex optimization method, dual Lagrangian
139   decomposition, and ShenJing Formula. Another study by Zhong et al. (2021) discussed a caching
140   strategy framework to optimize traffic load, and QoS in Multi-access Edge Computing named
141   GenCOSCO. The aim was to minimize the task execution time as a Mixed Integer Non-Linear
142   Programming (MINLP) optimization problem by considering the heterogeneity of task requests,
143   pre-storage of application data, and cooperation of the base station variables. The GenCOSCO was
144   used to propose the FixCS algorithm and was validated by calculating the average latency.
145   Moreover, Peng et al. (2021) proposed an application paradigm as a service chain for detailed
146   data offloading and location caching mechanisms. Each service chain was limited by leasing costs
147   and designed according to the queuing theory in computer networks. The research was further
148   validated by calculating the average response delay towards increasing cache server time and
149   capacity.
150   Wang et al. (2019) also studied the cached data offloading optimization on the edge network to
151   reduce requests made to the origin server for the same data. The focus was on the cache capacity
152   on the limited edge network, which means each cached data needed to be prioritized. The
153   optimization problem was proposed using a knapsack problem. This research used cyclic ACO-
154   GA with three variables: access count, access time, and data size. Research by Wang et al. (2019)
155   is used as the baseline in our study because, based on our manual calculations, there is an
156   opportunity to improve the value of the formulated objective function. The previously generated
157   objective function value is 0.4374, but this value can be even more optimal up to 0.4350. This is a
158   strong basis for us to improve the performance of CGACA.
159

Evolutionary Algorithms

1. Genetic Algorithm (GA)

GA was first introduced by John Holland in a publication entitled Adaptation in Natural and Artificial Systems Holland (1992) and observed to have adopted several phenomena such as natural selection and mutation for survival. GA used these adaptation principles to improve solutions in each generation Purnomo (2014). One other main principle is a crossover, a cross-breeding mechanism between two individuals (parents) to produce better quality offspring than both parents. It illustrates in Figure 2, where two individual chromosomes (parents) create new offspring working in groups with other genes to form a chromosome. This chromosome represents a solution vector from the completed case study, with each gene can mutate based on a certain probability, as illustrated in Figure 3.

Meanwhile, the quality of the genetic algorithm solution vector was measured by a fitness value. The highest value among the existing chromosomes is selected when the genetic algorithm finds the maximum optimization value. This fitness value was, calculated using the following equation (1). GA also uses elitism to maintain one or several of the best individuals in the next generation to produce other individuals with better fitness values Santosa and Ai (2017).

$$fitness = \frac{1}{f_{(x)} + \varepsilon} \qquad (1)$$

2. Ant Colony Optimization (ACO)

Dorigo first introduced ACO to find the shortest path in the Travelling Salesman Problem case study (TSP) (Dorigo, Maniezzo, and Colorni 1996). This algorithm adopts the behavior of ant colonies from the nest to the food source. Each ant can leave a pheromone trail on each of its paths. The pheromone substances are from endocrine glands, which can identify fellow ants or their groups by serving as a solid signal to influence other ants to follow in the footsteps of this pheromone. At first, each ant can determine its route and leave a pheromone trail to identify others, as indicated in Figure 4.

Moreover, the pheromone can evaporate, which means its path is lost faster on longer routes than shorter ones. It is difficult for other ants to follow the path on a longer route due to the evaporation of the pheromone trail. In comparison, the shorter route still has traces even though the first pheromone has disappeared, but it regains the path through the pheromone from other ants. The stronger signal attracts other ants to follow the same trail, as shown in Figure 5. An ant k at node r can select a route s(i,j) based on a certain probability, and those that have completed a route leaves a pheromone trail (τ) of $\tau_{i,j} \leftarrow \tau_{i,j} + \Delta\tau^k$ while the vaporized pheromone is calculated using $\tau_{i,j} \leftarrow (1 - \rho)\tau_{i,j}$ where ρ is a predetermined constant.

3. Cyclic ACO-GA

The performance of the Cyclic ACO-GA (CGACA) proposed by Wang et al. (2019) was improvised in this study to improve its final result by obtaining a more optimal objective function value. It is important to note beforehand that CGACA conducts cyclical algorithmic exchange between ACO and GA with the ACO algorithm run in the first iteration. At the same

198 time, the next solution search process is implemented through GA. ACO is applied when GA
199 experiences stagnation in five successive iterations until the maximum iteration is achieved.
200 Moreover, The ACO algorithm in CGACA used the roulette wheel, which has an unfavorable
201 impact due to its production of random solutions and superior individuals (Moodi, Ghazvini, and
202 Moodi, 2021; Lipowski and Lipowska, 2012). However, the CGACA framework proposed by
203 Wang et al. (2019) does not provide settlement action in a situation where the GA does not
204 experience stagnancy. It means the GA does not have a full role in exploiting the solution to the
205 fullest.
206   The results of our initial research showed that the ACO algorithm takes the longest time when
207 compared to GA and binary Particle Swarm Optimization (BPSO) in completing the cached data
208 optimization. The worst-case scenario in CGACA will occur if the GA algorithm experiences a
209 solution stagnation at the beginning of the iteration so that ACO will be rerun until the iteration
210 is complete. Such a situation makes the ACO iteration dominate, so the total CGACA time
211 consumption becomes very large. Table 1 explains the important weakness points to be
212 considered in improving the CGACA performance.
213

Methodology
215 **1. GenACO**
216   GenACO is an optimization framework that we propose to improve the performance of
217 CGACA produced by Wang et al. (2019). GenACO was proposed using the constant r0 to
218 overcome the roulette wheel weakness in solution probabilities. The r0 value is used to balance
219 exploring candidate solutions in the early iterations and focuses on exploiting the optimum
220 solution at the end of the iteration. In addition, GenACO has two execution modes, namely
221 cyclic and non-cyclic. Both managed to improve the quality of the solution.
222   Table 2 describes the rationale of GenACO maintaining the cyclic ACO-GA hybrid method.
223 Figure 6 illustrates the step of the proposed hybrid GenACO. The ACO algorithm is needed to
224 generate the best initial population for GA. The r0 has been used in this step. If the GA has a
225 stagnant solution, the last ACO will run again until the iteration ends. Detail explanation about
226 r0 will be explained further in the next section.
227   The Knapsack problem is a classic optimization problem that can be solved using brute force
228 or greedy methods. However, the greedy method does not guarantee an optimal solution, while
229 the brute force has a very high time complexity of $O(2^n)$. Based on Figure 6, assuming n is
230 cached data, m is the number of GenACO ants, and Nc is the iteration used, then the complexity
231 of GenACO is $O(Nc*n*m)$. Consequently, the time complexity of GenACO is better than the
232 classical method.
233 ***a. Solution Probability***
234   The ACO algorithm plays an essential role in generating the solution required by the GA to
235 produce the best population. However, the disproportionate use of trace pheromones can cause
236 the best solution of equation (4) not always to appear. The ACO algorithm in GenACO no longer
237 uses the roulette wheel to select a solution from all the ants. It was replaced by adding r0 as a
238 constant to be compared with a random number r [0,1], such that when r0 > r, the ant selects the

239 cached data with the enormous pheromone value and selects the cached data randomly when
240 otherwise.

$$\tau_i(t) = w_1 * \frac{ct(i)}{total(ct)} + w_2 * \frac{fr(i)}{total(fr)} + w_3 * \frac{sz(i)}{total(sz)} \qquad (2)$$

241   The initialization of the pheromone value (τ) follows equation (2) by using three cached data
242 property values, which include the access count (ct), access time (fr), and data size (sz) with each
243 multiplied by its respective weight w1=0.3 w2=0.3 w3=0.4 Wang et al. (2019). Moreover, the
244 visibility function in equation (3) also influences the choice of solutions made by the ant colony.
245 Therefore, the probability of cached data is selected and entered into the cache server follows
246 equation (4).

$$\eta_i(t) = w_1 * \frac{ct(i) * fr(i)}{sz(i)} \qquad (3)$$

$$P_i(t) = \frac{\tau_i(t)^\alpha * \eta_i(t)^\beta}{\tau_i(t)^\alpha * \eta_i(t)^\beta + \tau_i(t)^\alpha * \eta_i(t)^\beta} \qquad (4)$$

247
248   The r0 constant value becomes very important in determining the direction of ACO algorithm
249 solution selection in GenACO. The simulation showed the r0 = 0.5 is suitable for placing in the
250 first iteration of ACO to make the GenACO explore as many potential solutions as possible and
251 not easily trapped on the local optimum in the initial iterations. In addition, r0 = 0.9 is planned to
252 be set in the second part of ACO when the GA experiences a solution stagnation. This part is
253 expected to focus more on exploiting the search for the optimum value. Based on experience, it
254 will be difficult for the random number r[0,1] to be greater than 0.9. Therefore setting this value
255 seems to force ACO always to choose the cached data with the most significant probability. The
256 use of r0 in GenACO is illustrated in Table 3.

257 ***b. Cyclic and non-cyclic GenACO***
258   GenACO proposed two modes, cyclic and non-cyclic, to improve CGACA performance. The
259 first mode is described in Table 3. In contrast, the second mode only runs the ACO algorithm
260 once in the first iteration, after which the GA algorithm continues fully up to the maximum
261 iteration. The non-cyclic mode does not use r0 = 0.9 to avoid the local optimum in the initial
262 iteration. Therefore we used r0 = 0.7, which is expected to balance the exploration and
263 exploitation of solutions in the first iteration, as illustrated in Table 4.
264
265 **2. Cached-data representation**
266   The ACO and GA algorithms have different solution vector representations in modeling
267 cached data. In the GA algorithm, a chromosome creates as much array space as the total data,
268 with each representing the 0/1 condition of one cached data. A value of 1 indicates that the
269 selected cached data will be entered into the cache server. At the same time, 0 means it will not
270 be included. Meanwhile, in the ACO algorithm, the array space only consists of several routes
271 selected by the ants. One ant and another may have a different number of selected routes.

272   Moreover, the route chosen by this ant represents a collection of cached data to be entered
273   into the cache server, and the ACO solution representation on GenACO duplicates how ACO
274   works in solving TSP. Figure 7 and Figure 8 illustrate an example solution of GA and ACO,
275   respectively.
276

277   **3. Multi-objective function**
278   GenACO can solve the cached data offloading optimization problem using the knapsack
279   problem approach as a multi-objective optimization model. Its multi-objective optimization is
280   built using three property values which include access count (ct), access time (fr), and data size
281   (sz) contained in each cached data. However, not all data can enter the cache server due to the
282   limited capacity. Therefore, their priority value was determined based on these three property
283   values.
284   These three variables also have their respective objective functions. The objective function of
285   access count is in line with equation (8); access time with equation (9); and data size with
286   equation (10) Wang et al. (2019), multiplied by their respective weights $w_1 = w_2 = 0.3$, and w_3
287     $= 0.4$, which is then calculated the final objective function as profit (Fx) following (11) Wang et
288   al. (2019).
289   The smaller Fx indicates high priority. The n dataset consists of cached data candidates of x_1,
290     $x_2, x_3, ..., x_j$ that can be entered by the cache server with capacity S and profit F_x. Meanwhile, each
291   x_j consists of access count (ct), access time (fr), and data size (sz). Furthermore, the GenACO is
292   expected to determine the solution vector x_j when $x_j = 1$ to calculate objective function. The
293   cached data is ignored when $x_j = 0$ and will not be included in the objective function calculation
294   following (11). Therefore problem definition following equation (7).

$$\sum_{j=1}^{n} profit(F_x)\, x_j \le S \tag{7}$$

$$f_{CT} = \frac{1}{n}\sum_{j=1}^{n} \frac{D_{ct_{(j)}} - D_{ct_{(min)}}}{D_{ct_{(max)}} - D_{ct_{(min)}}} \tag{8}$$

$$f_{FR} = \frac{1}{n}\sum_{j=1}^{n} \frac{D_{fr_{(j)}} - D_{fr_{(min)}}}{D_{fr_{(max)}} - D_{fr_{(min)}}} \tag{9}$$

$$f_{SZ} = \frac{1}{n}\sum_{j=1}^{n} \frac{D_{sz_{(j)}} - D_{sz_{(min)}}}{D_{sz_{(max)}} - D_{sz_{(min)}}} \tag{10}$$

$$F_x = w_1 * (1 - f_{CT}) + w_2 * (1 - f_{FR}) + w_3 * f_{SZ} \tag{11}$$

295

296   **4. Performance measurement**
297   Evolutionary algorithms such as ACO and GA are widely used to complete optimization
298   work. Generally, the optimization performance measurement through the continuous domain is
299   usually measured by the optimum value of the objective function and the solution convergence

300 time achieved (Sahu, Panigrahi, and Pattnaik, 2012; Sun, Song, and Chen 2019). Meanwhile, the
301 knapsack problem belongs to a discrete domain. Therefore, additional performance indicators are
302 the best and worst profit, the average profit achieved, and the total number of items in the
303 knapsack (Rizk-Allah and Hassanien, 2018; Y. Li et al, 2020; Liu 2020). However, the best
304 solution in the cached data offloading optimization case study was selected based on the highest
305 number of knapsack items with the lowest objective function value.
306

Experiments and Evaluation

308 **1. Dataset**
309   GenACO was tested by datasets used in Wang et al. (2019) with three cached data property :
310 access count (ct), access time (fr), and data size (sz) with their respective sizes in Mbit, decimal,
311 and second. Moreover, the maximum capacity of the cache server was 950 Mbit. The dataset can
312 be accessed through the given GitHub repository.
313

314 **2. Simulation setup**
315   The single ACO and GA scenario was divided into four scenarios. Moreover, the CGACA
316 scenario was repeated three times to observe the solution stagnation position experienced by GA.
317 The results were compared with the proposed cyclic and non-cyclic GenACO. The simulation
318 was conducted using 10 particles and 20 iterations with the parameter settings of the six
319 scenarios presented in Table 5. Both parameters were adopted from Wang et al. (2019) then
320 simplified because the best solution average has been found in such particles and iterations
321 range. The simulation was conducted using PHP programming language and Mysql database to
322 facilitate our subsequent research in implementing fog computing architecture.
323

324 **3. Results and discussion**
325 ***a.   Performance comparison of single AC***
326   The complete simulation results in this paper can be seen in the Appendix at the end of this
327 paper. Table 6 showed the comparison of cached data optimization results for the single ACO
328 algorithm. According to Wang et al. (2019), the best known objective function (Fx) for cached
329 data offloading is 0.4374 with an optimum cached data (ch.dt) of 15. The first part of Table 6
330 showed the ACO algorithm from the CGACA section, and scenario-1a was observed to have
331 relied on the roulette wheel on the cached data selection mechanism to be loaded in cache
332 storage. However, this roulette wheel mechanism has two weaknesses (i) produce random
333 solutions (ii) cause the emergence of superior individuals. The average of Fx and cached data
334 generated in scenario-1a are the worst. The solution looks random and lacks convergence
335   Scenarios: 1b, 1c, and 1d are single ACO obtained from parts of the GenACO framework and
336 observed improvement in solution probability as described in section 3.1. The solution
337 probability based on r0 was compared with a random number r[0,1]. If r0 > r, then the solution
338 will follow (6), otherwise it will follow (5). Scenario-1b was set using r0=0.3 based on the
339 assumption that the random number r will be easier to be greater than this value, and the scenario
340 is expected to have more variety of solutions. However, scenario-1b managed to obtain an

341   optimal solution of 15 cached 7 times out of 20 iterations. The exciting thing from these results is
342   that the optimal solution was generated from different Fx values, 0.4374, 0.4405, and 0.4382.
343   The best solution was the smallest was 0.4374.
344   Scenario-1c was set using r0=0.5 and managed to obtain an optimal solution of 15 cached 5
345   times out of 20 iterations. Initially, r0=0.5 was assumed to have the capacity to produce more
346   optimal cached data than r0=0.3, but it did not. It proves unreliable to have the random r value in
347   a position more than or less than the r0 value. Meanwhile, the five best optimal solutions found
348   in this scenario have the same Fx value of 0.4382, and this means all the 15 selected cached data
349   have the same id_data. In this case, the best solution in scenario-1b is better than scenario-1c due
350   to its smaller Fx value.
351   The last single ACO, scenario-1d, has the highest value of 0.7 than the previous two
352   scenarios. The solution selection is expected to lead more to equation (6). Meanwhile, Table 6
353   showed this scenario-1d produces the best solution compared to the previous two scenarios by
354   having 95% optimal solution with 19 out of the 20 iterations having 15 cached data with two
355   dominant Fx values, which are 0.435 and 0.4405. It is important to note that the Fx value of
356   0.435 became the best value for the cached data offloading objective function generated from the
357   overall single ACO simulation scenario. The results of this simulation are in accordance with our
358   manual calculations to improve the results of previous research conducted by Wang et al. (2019)
359
360   **b. *Performance comparison of single GA***
361   Table 7 compared the single GA algorithm with four different mutation probability scenarios
362   and crossover compositions. The first part of the single GA simulation was observed to have
363   used a crossover composition of Parent-1 (P1)=15% and Parent-2 (P2)=85%. It means P1 gave
364   the first gene to the third gene, and P2 gave the fourth gene to the twentieth gene when the
365   crossover occurred. This calculation was used in all crossover scenarios in this single GA
366   simulation. Another critical parameter is gene mutation probability. The probability of gene
367   mutation in the simulation was 0.5, expecting that the crossover mechanism will not create a new
368   fitness value smaller than the previous best value.
369   Scenario-2a showed the worst solution among all the single GA simulation scenarios, with the
370   best solution produced having only 11 cached data with the same Fx values of 0.4653. At first
371   glance, the solution seems to be rapidly converging but quickly trapped in the local optimum due
372   to the inability of the crossover mechanism to produce a variety of solutions. Meanwhile,
373   scenario-2b managed to determine a (near) optimal solution with 14 cached data starting from
374   the 13th to the maximum iteration, and the search for solutions was varied. Moreover, its higher
375   mutation probability value compared with the previous scenario caused each gene in the
376   chromosome to have more flexibility to improve the solution quality in the next generation.
377   Scenario-2c uses the same mutation probability value of 0.25 as scenario-2b but has a
378   crossover composition of P1 and P2, slightly different from scenario-2b. Therefore, the results
379   also differ significantly from the average Fx value and the cached data produced. The last
380   scenario-2d used a mutation probability value of 0.5, with P1 and P2 having 50% composition
381   each. This scenario is generally similar to scenario-2a by being a fast solution towards

382 convergence, but it was quickly trapped at the optimum local value at the beginning of the
383 iteration. Moreover, the solution improvement with a mutation probability value of 0.5 also
384 failed to show, and this means the parameter setting in GA greatly affected the quality of the
385 solution produced.
386   Table 7 has shown that some of the best solutions produced by a single GA seem to be more
387 optimal (with a smaller Fx) than the best known Fx. However, this smaller Fx value does not
388 indicate a better solution because the cached data (ch.dt) is less than the best-known cached data.
389 Based on the best solution to the knapsack problem approach shown in equation (7), the single
390 GA solution has not outperformed the best-known solution.
391   The cached data offloading solution produced by GA is no better than ACO. None of the
392 solutions from GA achieved the best-known value. GA does not guarantee the search for a (near)
393 optimum solution as formulated by the pheromone function in ACO. However, GA has a
394 significant role in accelerating the convergence of cached data offloading case study solutions. If
395 correlated between Figure 6 Table 7 and Table 8, ACO and GA have a vital function in their
396 respective positions. In scenario-1b and scenario-1d, the ACO algorithm can find the optimal
397 solution in less than five iterations. This initial solution from ACO can be used as an outstanding
398 initial population for the GA algorithm to continue the solution search process until the
399 maximum iteration. This is supported by Table 7 and Table 8, which show the trend of solution
400 improvement from beginning to end.
401
402 *c. Performance comparison of CGACA*
403   Table 8 showed the simulation results for the CGACA framework with the ACO algorithm
404 run in the first iteration to generate the initial population. At the same time, the GA continued the
405 process of determining the solution. However, the GA algorithm was expected to experience
406 solution stagnation, which led to the application of the ACO algorithm for the second time to
407 continue the solution search process up to the maximum iteration. In the first part, the GA
408 solution was observed to have stagnated in the fifth iteration, and the ACO continued the search
409 process from the sixth iteration to the maximum. Therefore, this part of the experiment was the
410 worst case of CGACA since it required the longest execution time. It was associated with the
411 quick stagnation of GA, which led to the use of ACO to complete the solution search process
412 until the maximum iteration was reached.
413   In the second part, the GA algorithm did not experience any solution stagnation. Therefore,
414 the required execution time was the best, but it could not determine an optimal solution since the
415 initial population created by ACO was not optimal. This is one of the disadvantages of the
416 roulette wheel. It is also important to note that selecting a solution based on the roulette wheel
417 was only in one iteration, and the optimal solution was not found. Moreover, the GA algorithm
418 has a solution stagnation at the end of the iteration, precisely in the 13th iteration. It means the
419 time consumption at the end of the CGACA simulation is not as much as the first part.
420   Meanwhile, the quality of the CGACA solution in the third part was the best, and the solution
421 improvement on the last-ACO was expected to be successfully conducted by producing a (near)
422 optimum solution of 14 cached data. The same was found in the first part of CGACA, but the Fx

423   value produced in the third part is better. Furthermore, the simulation results in Table 8 showed
424   that the CGACA total time consumption increased along with the number of iterations completed
425   by the ACO. A more significant portion of iterations run by ACO led to the generation of more
426   extended time consumption. It means ACO required a long execution time to update the
427   pheromone trace on each selected cached data by all ants in each iteration. Therefore, it is very
428   inefficient if it has to complete the solution search by a single ACO.
429

430   ***d.   Comparison with cyclic and non-cyclic GenACO***

431   Appendix Table 8 and Table 9 marks the solution from ACO with a dark color on the cell
432   background. Table 9 compared the performance of cached data offloading using the CGACA and
433   GenACO frameworks. The GenACO simulations were divided into cyclic and non-cyclic
434   scenarios. Both utilized ACO to run once in the first iteration while the GA continue the process
435   and the last ACO was prepared to help when the GA experiences a solution stagnation.
436   The first part showed the simulation result obtained from the CGACA framework, which was
437   observed to have succeeded in determining the optimum solution based on the best value of 15
438   cached data with Fx=0.4374, but this was only found once in the 18th iteration. The first part of
439   Table 9 showed that the GA stagnated at the 7th iteration, leading to the last ACO from the 8th
440   iteration to the maximum. In the end, CGACA required more than 40 seconds which is not too
441   different from the value in Table 8, and this higher total time was associated with the dominance
442   of execution by ACO.
443   The second part of Table 9 showed the results for the GenACO simulation with cyclic mode,
444   and the working principle was found to be precisely the same as CGACA (baseline). However,
445   the probability of a cached data selection solution was improved. The first ACO on cyclic
446   GenACO was set using r0=0.7 with the expectation of generating more (near) optimal solutions.
447   The results showed that the GA algorithm has succeeded in determining the optimum solution of
448   15 cached data with an Fx value of 0.4350 better than the best-known Fx value between the 2nd
449   and 12th iterations. The improvement is only 0.0024 but based on the raw data we tracked,
450   GenACO can accommodate more small cached data. The server utility is maximized because it
451   can enter more cached data in the future.
452   However, the solution produced from the 13th iteration was considered stagnant since the Fx
453   value is the exact five times in a row. It led to the application of the last ACO starting from the
454   14th up to the maximum iteration. The solution produced did not get better. It was discovered
455   that all ants could not determine the optimal solution previously obtained until the maximum
456   iteration was run. It means the last ACO was stuck at the local optimum because the 14 cached
457   data solutions added more pheromone concentration than the previous one, the 15 cached data
458   solutions. Therefore, the last ACO preferred to follow the solution of the 14 cached data. It was
459   influenced by setting r0 = 0.9 on the last ACO, which directed the ants to the path with the
460   strongest pheromone than the new path required to be explored.
461   The third part of Table 9 showed the results of the non-cyclic GenACO simulation, which
462   only ran ACO on the first iteration and the GA on the second iteration up to the end. The GA
463   was observed to have stagnated solutions for five consecutive iterations but continuously

464   searched for solutions up to the maximum iteration. Meanwhile, r0 = 0.9 was used because ACO
465   was used only once. The ants preferred the cached data with maximum probability, and several
466   (near) optimum solutions are expected to be obtained by ACO ants in this non-cyclic mode to
467   form the best initial population in the GA algorithm. The 3rd to the 5th iteration results showed
468   that the GA immediately found the optimal solution for the best known 15 cached data with
469   Fx=0.4350 and maintained this value convergently from the 10th to the maximum iteration.
470   Based on table 10, the proposed hybrid GenACO method, particularly non-cyclic mode, is
471   superior to single ACO, GA, and CGACA in obtaining an average Fx, best solution, the optimal
472   amount of cached data during iteration. Full results can be seen in the Appendix.
473
474   *e.   The impact of different r0*
475   Based on Table 11, the value of r0 has an essential role in producing a solution used as an
476   initial parent for the GA algorithm. At r0=0.3, the solutions produced by ACO vary widely.
477   Three experiments conducted at r0=0.3 resulted in three different Fx and ch.dt values. This is in
478   line with the first ACO goal in the GenACO framework, which is to focus on solution
479   exploration. However, the resulting solution with r0=0.3 is quite far from the target best-known
480   value, so that this will make GA work difficult. The results shown are also in the use of r0=0.5.
481   However, the quality of the solution of r0=0.5 is better than r0=0.3. In general, the two values of
482   r0 are still not close to the best-known value, and the resulting solution does not appear to be
483   convergent. This proved our hypothesis in the Solution Probability section: the smaller the value
484   of r0, the more the solution chosen by the ant colony refers to equation (5). The use of r0=0.7
485   and r0=0.9 resulted in better solution quality. Both r0 solutions begin to look convergent. The Fx
486   and ch.dt values are getting closer to the best-known values. But keep in mind that r0 = 0.7 or r0
487   = 0.9 is not recommended for use in the first ACO because it can reduce the opportunities for
488   solution exploration. Both are better used in the last ACO to narrow the search space to focus on
489   exploiting the best value towards the best-known. The impact of using r0 on the GenACO
490   framework can be seen again, as shown in Table 9. Based on Table 9, the ACO solution using
491   r0=0.7 helps GA find the optimum solution and achieve solution convergence.
492
493   *f.   Comparison of stagnant solution CGACA and GenACO*
494   Figure 9 showed the solution stagnation of the three ACO–GA hybrid simulations. The first
495   ACO on CGACA obtains an initial solution that is not very good. This situation impacts GA
496   performance which fails to improve the initial solution so that it is quickly trapped in the local
497   optimum. Moreover, the roulette wheel mechanism and the probability of crossover and mutation
498   cause GA to fail to improve this initial solution. In the end, GA stuck to the solution stagnation
499   from iterations 2 to 5. The GA algorithm, which stagnated the solution far from the best known,
500   caused the last ACO hard to improve the solution and achieve solution convergence. However,
501   the roulette wheel mechanism used by the last ACO at CGACA failed to fix this until the
502   maximum iteration.
503   The GA algorithm on Cyclic GenACO also suffers from a solution stagnation. However,
504   before solution stagnation occurred, GA had a better initial solution. GA successfully maximized

505 the initial solution of the first ACO to become an optimal solution. However, because the
506 optimal solution is equal in five consecutive iterations, it is considered a stagnation of the
507 solution. In the end, the last ACO on cyclic GenACO was executed, but the optimal solution
508 previously generated could not be maintained. We suspect this situation was caused by set
509 r0=0.9, so the solution fell out from the previous optimum. In future work, this will be our
510 concern so r0 in the last ACO can be more adaptive to the previous solution.

511 The use of r0=0.7 in the non-cyclic GenACO simulation succeeded in obtaining a better initial
512 solution than CGACA and cyclic GenACO. This situation makes GA job easier. In addition, the
513 proper parameter setting on the probability of mutation and crossover makes GA performance
514 more reliable. It can be seen in the performance of GA, which can maintain the optimal solution
515 until the maximum iteration.

516 Based on Figure 9, non-cyclic GenACO is the best solution for average solution quality and
517 total time consumption. However, GenACO cyclic and non-cyclic modes did not have a
518 significant difference in total time consumption. Non-cyclic GenACO is only 4.5 seconds ahead
519 of cyclic mode. Based on Table 9, the saving time obtained by GenACO can be calculated using
520 equation (12).

$$Saving_{time} = \frac{(CGACA\ exec.time - GenACO\ exec.time)}{CGACA\ exec.time} * 100\% \qquad (12)$$

521

522 Therefore, we calculated cyclic GenACO saving time consumption by up to 38%, while non-
523 cyclic GenACO saving it by 47%. Both GenACO modes can be accepted as a solution to the
524 problem of cached data offloading. The ACO algorithm is very appropriate to use in the first
525 iteration to create an excellent and reliable initial population, making it easier for GA to find the
526 optimal solution quickly and good average quality of the solution.
527

Conclusions

529 An edge computing framework must have a cached data offloading capability that needs to be
530 optimized due to the limited storage capacity and maximum possible profit. This paper proposed
531 improving the hybrid ACO-GA algorithm performance using a new GenACO framework with
532 cyclic and non-cyclic modes. The simulation results show that GenACO minimizes the objective
533 function of cached data optimization from 0.4374 to 0.4350. In addition, GenACO can also
534 outperform the quality of the solution in terms of the average objective function generated.
535 Moreover, the parameter setting in evolutionary algorithms plays an essential role in the overall
536 algorithm performance.

537 Based on the simulation results, non-cyclic GenACO is the best mode for solving cached data
538 offloading optimization. This mode can reduce time consumption by up to 47%. Our further
539 research tests GenACO by measuring the hit ratio, examining the impact of latency and response
540 time on the user side in an edge computing environment.
541

Appendix

543 Tabel 6

544   Tabel 7

545   Tabel 8

546   Tabel 9

547

References

549   Abraham Silberschatz, Peter B. Galvin, Greg Gagne. 2008. *Operating System Concepts*.
550   Reading, MA, USA: Addison Wesley.

551   Ali, Waleed, Siti Mariyam, and Abdul Samad. 2012. "Intelligent Web Proxy Caching
552   Approaches Based on Machine Learning Techniques." *Decision Support Systems*
553   53(3):565–79. doi: 10.1016/j.dss.2012.04.011.

554   Ali, Waleed, Siti Mariyam Shamsuddin, and Abdul Samad Ismail. 2011. "A Survey of Web
555   Caching and Prefetching." *International Journal Advance. Soft Comput. Appl.* 3(1):1–27.

556   Ali, Waleed, Siti Mariyam Shamsuddin, and Abdul Samad Ismail. 2012. "Intelligent Naïve
557   Bayes-Based Approaches for Web Proxy Caching." *Knowledge-Based Systems* 31:162–75.
558   doi: 10.1016/j.knosys.2012.02.015.

559   Azlan, Che Ahmad, Jeannie Hsiu Ding Wong, Li Kuo Tan, Muhammad Shahrun Nizam A.D.
560   Huri, Ngie Min Ung, Vinod Pallath, Christina Phoay Lay Tan, Chai Hong Yeong, and
561   Kwan Hoong Ng. 2020. "Teaching and Learning of Postgraduate Medical Physics Using
562   Internet-Based e-Learning during the COVID-19 Pandemic – A Case Study from
563   Malaysia." *Physica Medica* 80(October):10–16. doi: 10.1016/j.ejmp.2020.10.002.

564   Bala, Mohammad Irfan, and Mohammad Ahsan Chishti. 2019. "Survey of Applications,
565   Challenges and Opportunities in Fog Computing." *International Journal of Pervasive*
566   *Computing and Communications* 15(2):80–96. doi: 10.1108/IJPCC-06-2019-059.

567   Carvalho, Gonçalo, Bruno Cabral, Vasco Pereira, and Jorge Bernardino. 2020. "Computation
568   Offloading in Edge Computing Environments Using Artificial Intelligence Techniques."
569   *Engineering Applications of Artificial Intelligence* 95(July):103840. doi:
570   10.1016/j.engappai.2020.103840.

571   De', Rahul, Neena Pandey, and Abhipsa Pal. 2020. "Impact of Digital Surge during Covid-19
572   Pandemic: A Viewpoint on Research and Practice." *International Journal of Information*
573   *Management* 55(June):102171. doi: 10.1016/j.ijinfomgt.2020.102171.

574   Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. 1996. "Ant System: Optimization by a
575   Colony of Cooperating Agents." *IEEE Transactions on Systems, Man and Cybernetics, Part*
576   *B (Cybernetics)* 26(1):29–41. doi: 10.1109/3477.484436.

577   Dutta, Kaushik, and Debra Vandermeer. 2017. "Caching to Reduce Mobile App Energy
578   Consumption." *ACM Transactions on the Web* 12(1):1–30. doi:
579   https://doi.org/10.1145/3125778.

580   Elgendy, Ibrahim A., Weizhe Zhang, Yu-Chu Tian, and Keqin Li. 2019. "Resource Allocation
581   and Computation Offloading with Data Security for Mobile Edge Computing." *Future*
582   *Generation Computer Systems* 100:531–41. doi: 10.1016/j.future.2019.05.037.

583   Ertenlice, Okkes, and Can B. Kalayci. 2018. "A Survey of Swarm Intelligence for Portfolio
584   Optimization: Algorithms and Applications." *Swarm and Evolutionary Computation*
585   39(December 2016):36–52. doi: 10.1016/j.swevo.2018.01.009.

586   Ha, Kiryong, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev
587   Satyanarayanan. 2014. "Towards Wearable Cognitive Assistance." Pp. 68–81 in
588   *Proceedings of the 12th annual international conference on Mobile systems, applications,*
589   *and services*. New York, NY, USA: ACM.

590 Haman, Michael. 2020. "The Use of Twitter by State Leaders and Its Impact on the Public
591   during the COVID-19 Pandemic." *Heliyon* 6(11):e05540. doi:
592   10.1016/j.heliyon.2020.e05540.
593 Holland, John H. 1992. *Adaptation in Natural and Artificial Systems*.
594 Huang, Liang, Xu Feng, Cheng Zhang, Liping Qian, and Yuan Wu. 2019. "Deep Reinforcement
595   Learning-Based Joint Task Offloading and Bandwidth Allocation for Multi-User Mobile
596   Edge Computing." *Digital Communications and Networks* 5(1):10–17. doi:
597   10.1016/j.dcan.2018.10.003.
598 Kuang, Zhufang, Zhihao Ma, Zhe Li, and Xiaoheng Deng. 2021. "Cooperative Computation
599   Offloading and Resource Allocation for Delay Minimization in Mobile Edge Computing."
600   *Journal of Systems Architecture* 118(July 2020):102167. doi:
601   10.1016/j.sysarc.2021.102167.
602 Li, Chunlin, Mingyang Song, Shaofeng Du, Xiaohai Wang, Min Zhang, and Youlong Luo. 2020.
603   "Adaptive Priority-Based Cache Replacement and Prediction-Based Cache Prefetching in
604   Edge Computing Environment." *Journal of Network and Computer Applications*
605   165:102715. doi: 10.1016/j.jnca.2020.102715.
606 Li, Ya, Yichao He, Xuejing Liu, Xiaohu Guo, and Zewen Li. 2020. "A Novel Discrete Whale
607   Optimization Algorithm for Solving Knapsack Problems." *Applied Intelligence*
608   50(10):3350–66. doi: 10.1007/s10489-020-01722-3.
609 Lipowski, Adam, and Dorota Lipowska. 2012. "Roulette-Wheel Selection via Stochastic
610   Acceptance." *Physica A: Statistical Mechanics and Its Applications* 391(6):2193–96. doi:
611   10.1016/j.physa.2011.12.004.
612 Liu, Zhuo. 2020. "An Analysis of Particle Swarm Optimization of Multi-Objective Knapsack
613   Problem." *ICITM 2020 - 2020 9th International Conference on Industrial Technology and
614   Management* 302–6. doi: 10.1109/ICITM48982.2020.9080345.
615 Luo, Zhaohui, Minghui LiWang, Zhijian Lin, Lianfen Huang, Xiaojiang Du, and Mohsen
616   Guizani. 2017. "Energy-Efficient Caching for Mobile Edge Computing in 5G Networks."
617   *Applied Sciences* 7(6):557. doi: 10.3390/app7060557.
618 Mertz, Jhonny, and Ingrid Nunes. 2017. "A Qualitative Study of Application-Level Caching."
619   *IEEE Transactions on Software Engineering* 43(9):798–816. doi:
620   10.1109/TSE.2016.2633992.
621 Mertz, Jhonny, and Ingrid Nunes. 2018. "Automation of Application-Level Caching in a
622   Seamless Way." *Software: Practice and Experience* 48(6):1218–37. doi: 10.1002/spe.2571.
623 Moodi, Mahdi, Mahdieh Ghazvini, and Hossein Moodi. 2021. "A Hybrid Intelligent Approach to
624   Detect Android Botnet Using Smart Self-Adaptive Learning-Based PSO-SVM."
625   *Knowledge-Based Systems* 222:106988. doi: 10.1016/j.knosys.2021.106988.
626 Naeem, Muhammad, and Wilson Ozuem. 2021. "The Role of Social Media in Internet Banking
627   Transition during COVID-19 Pandemic: Using Multiple Methods and Sources in
628   Qualitative Research." *Journal of Retailing and Consumer Services* 60(January):102483.
629   doi: 10.1016/j.jretconser.2021.102483.
630 Nimrod, Galit. 2020. "Changes in Internet Use When Coping With Stress: Older Adults During
631   the COVID-19 Pandemic." *The American Journal of Geriatric Psychiatry* 28(10):1020–24.
632   doi: 10.1016/j.jagp.2020.07.010.
633 Peng, Kai, Jiangtian Nie, Neeraj Kumar, Chao Cai, Jiawen Kang, Zehui Xiong, and Yang Zhang.
634   2021. "Joint Optimization of Service Chain Caching and Task Offloading in Mobile Edge
635   Computing." *Applied Soft Computing* 103:107142. doi: 10.1016/j.asoc.2021.107142.

636   Pescosolido, Loreto, Marco Conti, and Andrea Passarella. 2019. "D2D Data Offloading in
637   Vehicular Environments with Optimal Delivery Time Selection." *Computer*
638   *Communications* 146(July):63–84. doi: 10.1016/j.comcom.2019.07.016.

639   Prerna, Divya, Rajkumar Tekchandani, and Neeraj Kumar. 2020. "Device-to-Device Content
640   Caching Techniques in 5G: A Taxonomy, Solutions, and Challenges." *Computer*
641   *Communications* 153(November 2019):48–84. doi: 10.1016/j.comcom.2020.01.057.

642   Purnomo, Hindriyanto Dwi. 2014. *Belajar Metode Optimasi Metaheuristik Menggunakan*
643   *Matlab*. Yogyakarta: Gava Media.

644   Rizk-Allah, Rizk M., and Aboul Ella Hassanien. 2018. "New Binary Bat Algorithm for Solving
645   0–1 Knapsack Problem." *Complex & Intelligent Systems* 4(1):31–53. doi: 10.1007/s40747-
646   017-0050-z.

647   Sahu, Amaresh, Sushanta Kumar Panigrahi, and Sabyasachi Pattnaik. 2012. "Fast Convergence
648   Particle Swarm Optimization for Functions Optimization." *Procedia Technology* 4:319–24.
649   doi: 10.1016/j.protcy.2012.05.048.

650   Sai, Ying, Dong-zhu Fan, and Meng-yang Fan. 2020. "Cooperative and Efficient Content
651   Caching and Distribution Mechanism in 5G Network." *Computer Communications*
652   161(July):183–90. doi: 10.1016/j.comcom.2020.07.030.

653   Santosa, Budi, and The Jin Ai. 2017. *Pengantar Metaheuristik Implementasi Dengan Matlab*.
654   Edisi Pert. Surabaya: ITS Tekno Sains.

655   Shi, Weisong, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. "Edge Computing:
656   Vision and Challenges." *IEEE Internet of Things Journal* 3(5):637–46. doi:
657   10.1109/JIOT.2016.2579198.

658   Sun, Lijun, Xiaodong Song, and Tianfei Chen. 2019. "An Improved Convergence Particle
659   Swarm Optimization Algorithm with Random Sampling of Control Parameters." *Journal of*
660   *Control Science and Engineering* 2019:1–11. doi: 10.1155/2019/7478498.

661   Tatar, Alexandru, Marcelo Dias de Amorim, Serge Fdida, and Panayotis Antoniadis. 2014. "A
662   Survey on Predicting the Popularity of Web Content." *Journal of Internet Services and*
663   *Applications* 5(1):1–20. doi: 10.1186/s13174-014-0008-y.

664   Tian, Geng, and Michael Liebelt. 2014. "An Effectiveness-Based Adaptive Cache Replacement
665   Policy." *Microprocessors and Microsystems* 38(1):98–111. doi:
666   10.1016/j.micpro.2013.11.011.

667   Wang, Danyue, Xingshuo An, Xianwei Zhou, and Xing Lü. 2019. "Data Cache Optimization
668   Model Based on Cyclic Genetic Ant Colony Algorithm in Edge Computing Environment."
669   *International Journal of Distributed Sensor Networks* 15(8):1–13. doi:
670   10.1177/1550147719867864.

671   Xu, Xiaolong, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, and Lianyong
672   Qi. 2019. "A Computation Offloading Method over Big Data for IoT-Enabled Cloud-Edge
673   Computing." *Future Generation Computer Systems* 95:522–33. doi:
674   10.1016/j.future.2018.12.055.

675   Zhang, Changhao. 2020. "Design and Application of Fog Computing and Internet of Things
676   Service Platform for Smart City." *Future Generation Computer Systems* 112:630–40. doi:
677   10.1016/j.future.2020.06.016.

678   Zhao, Xianlong, Kexin Yang, Qimei Chen, Duo Peng, Hao Jiang, Xianze Xu, and Xinzhuo
679   Shuang. 2019. "Deep Learning Based Mobile Data Offloading in Mobile Edge Computing
680   Systems." *Future Generation Computer Systems* 99:346–55. doi:
681   10.1016/j.future.2019.04.039.

682   Zhong, Shijie, Songtao Guo, Hongyan Yu, and Quyuan Wang. 2021. "Cooperative Service
683   Caching and Computation Offloading in Multi-Access Edge Computing." *Computer*
684   *Networks* 189(January):107916. doi: 10.1016/j.comnet.2021.107916.
685   Zhu, Yuhao, and Vijay Janapa Reddi. 2017. "Optimizing General-Purpose CPUs for Energy-
686   Efficient Mobile Web Computing." *ACM Transactions on Computer Systems* 35(1). doi:
687   10.1145/3041024.
688   Zulfa, Mulki Indana, Rudy Hartanto, and Adhistya Erna Permanasari. 2020. "Caching Strategy
689   for Web Application – a Systematic Literature Review." *International Journal of Web*
690   *Information Systems* 16(5):545–69. doi: 10.1108/IJWIS-06-2020-0032.
691

Figure 1

Fog computing environment

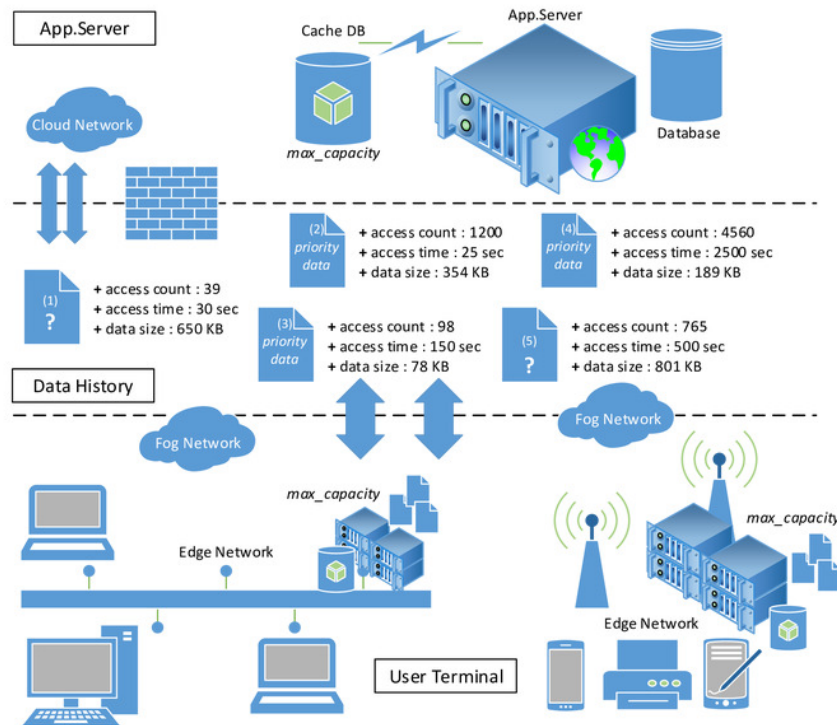



Figure 2

Crossover mechanism of genetic algorithm

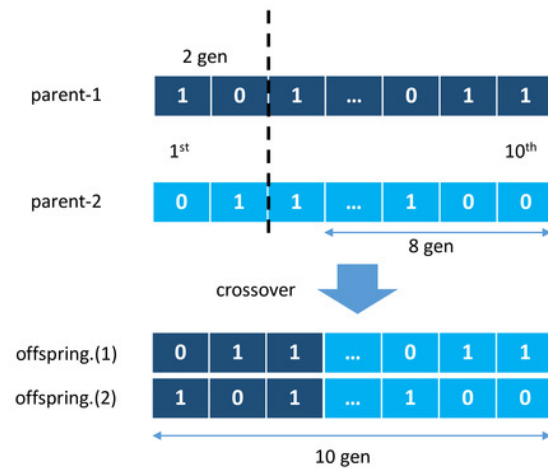



Figure 3

Examples of mutations of genetic algorithm

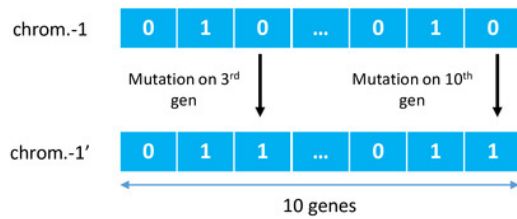



Figure 4

Initial conditions between ant colony in a nest and food source (Dorigo et al, 1996)

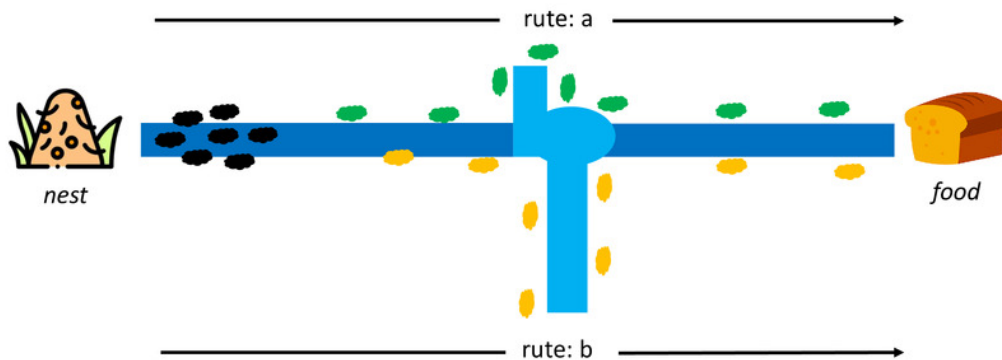



Figure 5

Rute-a as the shortest path from a nest to food source by ant colony (Dorigo et al, 1996)

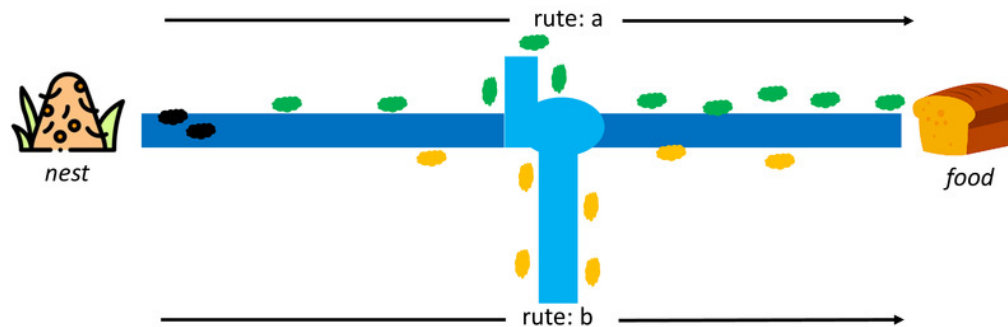



Figure 6

The step of the proposed hybrid GenACO

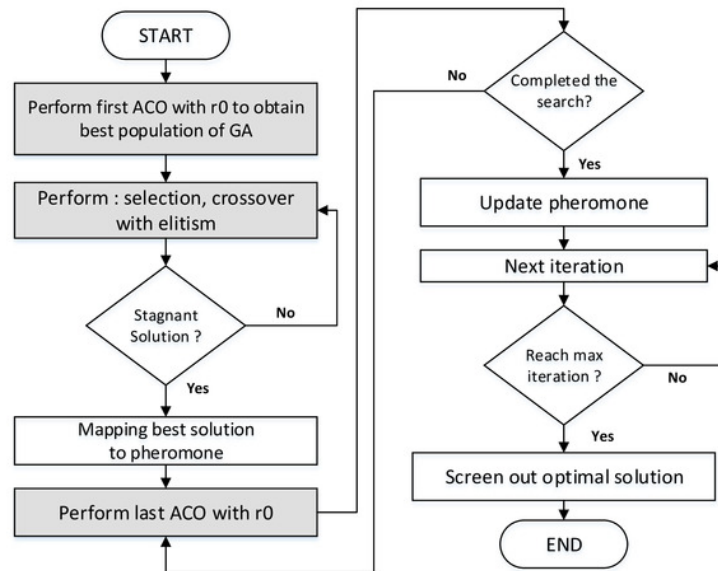



Figure 7

An example of solution vector in GA algorithm

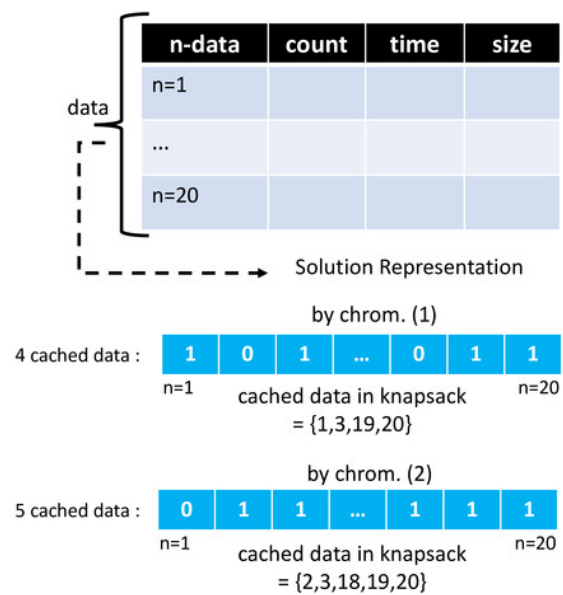



Figure 8

An example of solution vector in ACO algorithm

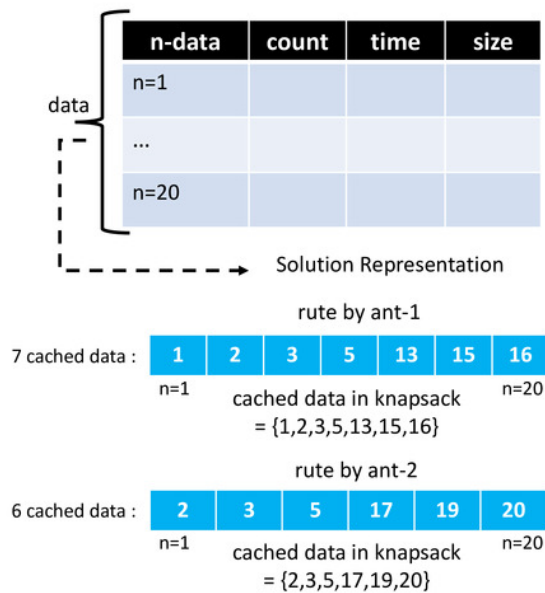
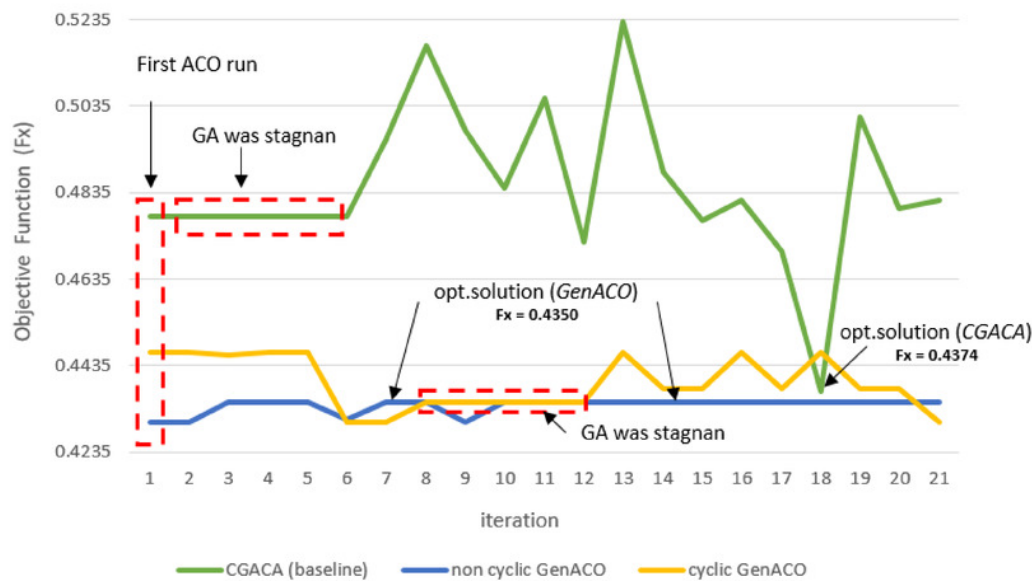



Figure 9

Objective function (Fx) comparison between CGACA and GenACO

The weaknesses of CGACA framework

No	Aspect	Explanation of weakness
1	Time consuming	In worst case condition, CGACA takes a very long time because the dominant iteration is held by ACO.
2	Dominant iteration	The solution search process is completed by ACO over a very long time when the GA experiences a solution stagnation condition at the beginning of the iteration.
3	GA iteration limitation	The CGACA framework limits the GA to a maximum of 20 iterations from a total of 100 allowed for the ACO. This limits the optimal performance of the GA.
4	Solution probability	The ACO algorithm in the CGACA framework uses a roulette wheel mechanism for solution selection and this has an unfavorable impact due to its production of random solutions and superior individuals (Moodi, Ghazvini, and Moodi, 2021; Lipowski and Lipowska, 2012).

The advantages of ACO-GA hybrid method

Algorithm	Aspect	Advantages
ACO	Initial solution	GenACO used the initial population by ACO which was not created randomly but followed pheromone trail based on a particular objective function.
	Solution probability	GenACO does not use a roulette wheel to select a solution from each ant. ACO needs to guarantee as many (near) optimum solutions as possible due to the fact that is just being run for the first time. The use of r0 will be used here.
GA	Avoiding local optimum	GA is more advantageous in the mechanism of selection, crossover, and mutation which create a reliable algorithm performance to avoid optimal local traps.
	Elitism	GenACO implements the elitism principle of the GA algorithm to improve the solution in each subsequent iteration.

Illustration on the use of r0 in the three cyclic GenACO steps

1	First ACO	r0 = 0.5	condition: if r0 > random r
			solution: random ($P_i(t)$) (5)
			assumption: it is easier for the random number r to be greater than 0.5
2	Genetic Algorithm		
3	Last ACO	r0 = 0.9	condition: if r0 > random r
			solution: maximum ($P_i(t)$) (6)

Pseudo-code

```
Input: ant m, iteration ith, cached_data n, visibility_matrix v(n),
probability P(n,v), server_capacity S
Output: Knapsack_items Kp, Obj_Func Fx

% initialization
foreach n=1: all cached_data do
     v(n) = eq.(3)
end for

while ith < ith_max do
  while S(Kp) < S_max do
    if ith == 1 then
    % run first ACO (Table 3: step-1)
       foreach m=1: all ants do
          foreach n=1: all cached_data do
             calculate P(n,v); generated random r[0,1]; r0=0.5;
             if (r0 > r) Kp(n) = eq.(5); calculate S(Kp); calculate Fx(S);
          end for
       end for
    end if
    if ith > 1 then
    %run GA (Table 3: step-2)
       Run crossover and mutation (Table 5)
       calculate profit(Kp); Update S(Kp); update Fx(S);
       if GA has stagnan then
          %run last ACO (Table 3: step-3)
          foreach m=1: all ants do
             foreach n=1: all cached_data do
                calculate P(n,v); generated random r[0,1]; r0=0.9;
                if (r0 > r) Kp(n) = eq.(6); Update S(Kp); update Fx(S);
             end for
          end for
       end if
    end if
  end
  screenout(Kp); screenout(Fx);
end
```

Illustration on the use of r0 in the two GenACO non-cyclic steps

			condition: if r0 > random r
1	First ACO	r0 = 0.7	solution: random $(P_i(t))$ and maximum $(P_i(t))$
			assumption: a more balanced process of exploring and exploiting solutions
2	Genetic Algorithm		

Pseudo-code

```
Input: ant m, iteration ith, cached_data n, visibility_matrix v(n),
probability P(n,v), server_capacity S
Output: Knapsack_items Kp, Obj_Func Fx

% initialization
foreach n=1: all cached_data do
     v(n) = eq.(3)
end for

while ith < ith_max do
  while S(Kp) < S_max do
    if ith == 1 then
    % run first ACO (Table 3: step-1)
       foreach m=1: all ants do
          foreach n=1: all cached_data do
             calculate P(n,v); generated random r[0,1]; r0=0.7;
             if (r0 > r) Kp(n) = eq.(5);
             else Kp(n) = eq.(6);
             calculate S(Kp); calculate Fx(S);
          end for
       end for
    end if
    if ith > 1 then
    %run GA (Table 3: step-2)
       Run crossover and mutation (Table 5)
       calculate profit(Kp); Update S(Kp); update Fx(S);
    end if
  end
  screenout(Kp); screenout(Fx);
end
```

Parameters settings

Scenario	Algorithms	Parameters		
1	Single ACO (CGACA)	a)	$\alpha = 0.3; \beta = 0.5; \rho = 0.001$	
	Single ACO (GenACO)	b)	$\alpha = 0.3; \beta = 0.5; r_0 = 0.3; \rho = 0.001$	
		c)	$\alpha = 0.3; \beta = 0.5; r_0 = 0.5; \rho = 0.001$	
		d)	$\alpha = 0.3; \beta = 0.5; r_0 = 0.7; \rho = 0.001$	
2	Single GA	a)	prob.mut.= 0.12; crossover (P1 = 15%; P2 = 85%)	
		b)	prob.mut..= 0.25; crossover (P1 = 15%; P2 = 85%)	
		c)	prob.mut.= 0.25; crossover (P1 = 35%; P2 = 65%)	
		d)	prob.mut.= 0.5; crossover (P1 = 50%; P2 = 50%)	
3	CGACA	✓ First ACO: $\alpha = 0.3; \beta = 0.5; \rho = 0.001;$ ✓ GA: mutation = *random*; crossover = *roulette wheel* ✓ Last ACO: $\alpha = 0.3; \beta = 0.5; \rho = 0.001;$		
4	Cyclic GenACO	✓ First ACO: $\alpha = 0.3; \beta = 0.5; r_0 = 0.7; \rho = 0,001$ ✓ GA: Mutation probability = 0,12; crossover = P1:15%, P2:85% ✓ Last ACO: $\alpha = 0.3; \beta = 0.5; r_0 = 0.9; \rho = 0.001$		
5	Non cyclic GenACO	✓ First ACO: $\alpha = 0.3; \beta = 0.5; r_0 = 0.7; \rho = 0.001$ ✓ GA: Mutation probability = 0,25; crossover = P1:15%, P2:85%		

Cached data offloading performance comparison by single ACO

Single ACO								
	Scenario-1a		Scenario-1b		Scenario-1c		Scenario-1d	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
AVG	0.4916	12.24	0.4459	14.05	0.4429	13.67	0.4358	14.95
BEST	0.4533	14	0.4374	15	0.4382	15	0.435	15
WORST	0.4928	10	0.4708	12	0.4843	12	0.4305	14

Cached data offloading performance comparison by single GA

Single GA								
	Scenario-2a		Scenario-2b		Scenario-2c		Scenario-2d	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
AVG	0.4653	11	0.4012	12.95	0.3933	12.33	0.4054	13
BEST	0.4653	11	0.3855	14	0.3776	13	0.3999	13
WORST	0.4653	11	0.4105	10	0.4928	10	0.4063	13

Cached data offloading performance by CGACA

Framework : CGACA						
	stagnant: ith=5		**normal**		**stagnant:** ith=13	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
AVG	0.4904	12.3	0.5079	11.05	0.4851	12.75
BEST	0.4672	14	0.499	12	0.4476	14
WORST	0.5208	11	0.5169	7	0.4749	11
TIME	40.21 sec		31.29 sec		34.77 sec	

A comparison of cached data offloading performance achieved by CGACA and GenACO

th.	CGACA		cyclic GenACO		non-cyclic GenACO	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
AVG	0.4848	12.33	0.4395	14.14	0.4342	14.71
BEST	0.4374	15	0.435	15	0.435	15
WORST	0.5175	11	0.4458	12	0.4305	14
TIME	48.44 sec		25.34 sec		29.84 sec	

A comparison of the best results from single GA, ACO, CGACA, and GenACO

Best Scenario Comparison								
	Single GA		Single ACO		CGACA		GenACO	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
AVG	0.4012	12.95	0.4358	14.95	0.4851	12.75	0.4342	14.71
BEST	0.3855	14	0.435	15	0.4476	14	0.435	15
WORST	0.4105	10	0.4305	14	0.4749	11	0.4305	14

Solution comparison of di□erent r0

Comparing the r0								
Try	r0=0.3		r0=0.5		r0=0.7		r0=0.9	
	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt	Fx	ch.dt
1	0.4947	10	0.4408	12	0.4470	14	0.4355	13
2	0.4894	12	0.4727	12	0.4355	14	0.4355	14
3	0.4856	11	0.4575	13	0.4498	12	0.4355	14