

Cache Replacement Policies

This document contains the algorithms used for Cache Replacement in Edge Cloud.

- [LRU](#) - Least Recently Used
- [FIFO](#) - First In First Out
- [LFU](#) - Least Frequently Used
- [GDSF](#) - Greedy Dual Size Frequency
- [SCRCP](#) - Service Aware Cache Replacement Policy

Algorithm Image class

```
1: ▷ Class to create free & running set of images.
2: Image:
3:   String job_id                                ▷ Stores the job id of the service
4:   int frequency                                ▷ Used only in LFU & GDSF; Stores the frequency of a service
5:   int stay                                     ▷ Used only in SCRCP; stores the frequency of a service in the cache
6:   double priority                             ▷ Used only in GDSF; Stores the priority of a service
7:   double SCRCP                                ▷ Used only in SCRCP; Stores the SCRCP score of a service
8:   long timestamp                              ▷ Stores the timestamp of the service
9:   long task_index                             ▷ Stores the task index of the service
10:  double cpu                                  ▷ Stores the CPU requirement of the service
11:  double ram                                  ▷ Stores the RAM requirement of the service
12:  double runtime_disk                          ▷ Stores the runtime disk requirement of the service
13:  double storage                              ▷ Stores the amount of storage required by service to be cloned in cache
14:
15: Image(String job_id, long timestamp, long task_index, double cpu, double ram, double runtime_disk, double storage):
16:                                     ▷ Constructor to initialize instance of Image class
17:   this.variable = variable ∀ variables
18:
19: Image(String job_id, int frequency, long task_index, double cpu, double ram, double runtime_disk, double storage):
20:                                     ▷ Modified constructor for LFU
21:   this.variable = variable ∀ variables
22:
23: Image(String job_id, int frequency, double priority, long task_index, double cpu, double ram, double runtime_disk, double storage):
24:                                     ▷ Modified constructor for GDSF
25:   this.variable = variable ∀ variables
26: Image(String job_id, int frequency, int stay, double SCRCP, long task_index, double cpu, double ram, double runtime_disk, double
   storage):
27:                                     ▷ Modified constructor for SCRCP
28:   this.variable = variable ∀ variables
```

Algorithm Minimum Timestamp Finder

```
1: ▷ Function to find service id that has minimum timestamp from the set of images for LRU & FIFO
2: String min_timestamp(HashMap< String, Image > hm)
3:   String min_ts = ""
4:   long ts = long.MAX_VALUE
5:   hm.forEach((Service_id, Image) →                                     ▷ Iterating over whole set of images
6:     if Image.timestamp < ts then
7:       ts = Image.timestamp
8:       min_ts = service_id
9:     end if
10:   return min_ts                                     ▷ Service id having minimum(oldest) timestamp
```

Algorithm Cache class to store the cloud and service data

```
1: Cache:
2:   int no_services = 0
3:   double  $CPU_{max}$  = 1.00
4:   double  $RAM_{max}$  = 1.00
5:   double  $Storage_{max}$  = 1.00
6:   double  $Runtime\_disk_{max}$  = 1.00
7:   double  $Disk\_Storage_{max}$  = 2.00
8:   double  $CPU_{curr}$  = 0
9:   double  $RAM_{curr}$  = 0
10:  double  $Runtime\_disk_{curr}$  = 0
11:  double  $Storage_{curr}$  = 0
12:  Hashmap < String, Image > Running
13:  Hashmap < String, Image > Free
```

▷ no of services currently in the Edge Cache
▷ max CPU available in cache(scaled to 1)
▷ max RAM available in cache(scaled to 1)
▷ max storage available in cache(scaled to 1)
▷ max runtime disk available(scaled to 1)
▷ max sum of runtime disk and storage
▷ Current usage of CPU in Edge Server
▷ Current usage of RAM in Edge Server
▷ Current usage of runtime disk in Edge Server
▷ Current usage of Storage in cache
▷ Contains images of all running services and their service id
▷ Contains images of all free services and their service id

Algorithm Least Frequency Finder

```
1: ▷ Function to find service id that has least frequency from the set of images for LFU
2: String min_frequency(Hashmap< String, Image > hm)
3:   String min_freq = ""
4:   int freq = Integer.MAX_VALUE
5:   hm.forEach((Service_id, Image) →
6:     if Image.Frequency < freq then
7:       freq = Image.Frequency
8:       min_freq = Service_id
9:     end if
10:
11:   return min_freq
```

▷ Iterating over whole set of images
▷ Service id having least frequency

Algorithm Priority Calculator for GDSF

```
1: ▷ Function to calculate priority of a service for GDSF
2: double priority_calc(double clock, int frequency, double size)
3:   if size == 0 then
4:     return Double.MAX_VALUE
5:   end if
6:
7:   return ( $clock + \frac{frequency}{size}$ )
```

▷ Services consuming 0 disk resources

Algorithm Least Priority Finder

```
1: ▷ Function to find service id that has least priority from the set of images
2: String min_priority(Hashmap< String, Image > hm)
3:   String min_priority = ""
4:   double priority_min = Double.MAX_VALUE
5:   hm.forEach((Service_id, Image) →
6:     if Image.priority < priority_min then
7:       priority_min = Image.priority
8:       min_priority = Service_id
9:     end if
10:
11:   return min_priority
```

▷ Iterating over whole set of images
▷ Service id having least priority

Algorithm LRU Replacement in Edge Cloud

```
1: int cache_hit = 0 ▷ No of cache hit
2: while Cloud receives request do
3:   String[ ] tokens = request.split() ▷ Contains all information in the request
4:   String service_id = tokens[0] ▷ Generate service id for the service
5:   double CPUreq = tokens[3], RAMreq = tokens[4], Runtime_Diskreq = tokens[5] ▷ Service requirements
6:   double Storagereq = 0 ▷ Assuming it be 0 for now
7:   if Cache.Running.containsKey(service_id) then ▷ If the service is cached
8:     cache_hit ++
9:     Image to_update = Cache.Running.get(service_id)
10:    to_update.timestamp = current_timestamp ▷ Update its timestamp
11:    Cache.Running.put(service_id, to_update)
12:  else if Cache.Free.containsKey(service_id) then ▷ If the image of service is present in Free list
13:    Image replace = Cache.Free.get(service_id)
14:    if There are enough resources in the edge cloud then
15:      replace.timestamp = current_timestamp
16:      Cache.Running.put(service_id, replace) ▷ Adding it to the Running list
17:      Cache.CPUcurr += replace.CPU ▷ Updating current CPU status
18:      Cache.RAMcurr += replace.RAM ▷ Updating current RAM status
19:      Cache.Runtime_diskcurr += replace.Runtime_disk ▷ Updating current Runtime Disk status
20:      Cache.Storagecurr += replace.Storage ▷ Updating current storage status
21:      Cache.Free.remove(replace) ▷ Removing it from free list
22:    end if
23:  else ▷ If the service is not cached
24:    Running' ← Cache.Running
25:    Free' ← Cache.Free ▷ Creating copies. Update cloud only if enough resources are available
26:    CPU'curr ← Cache.CPUcurr
27:    RAM'curr ← Cache.RAMcurr ▷ Creating copies of Cache Status
28:    Runtime_disk'curr ← Cache.Runtime_diskcurr
29:    Storage'curr ← Cache.Storagecurr
30:    while There is no enough resources in edge cloud do
31:      if CPUreq + CPU'curr > Cache.CPUmax || RAMreq + RAM'curr > Cache.RAMmax then
32:        String to_remove = min_timestamp(Running') ▷ Finding service id with least time stamp
33:        Image remove = Running'.get(to_remove)
34:        Free'.put(to_remove, remove) ▷ Inserting its image to set of free images
35:        CPU'curr -= remove.CPU ▷ Updating current CPU status
36:        RAM'curr -= remove.RAM ▷ Updating current RAM status
37:        Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
38:        Running'.remove(to_remove) ▷ Removing its image from set of running images
39:      end if
40:      if Disk_Storagereq + Disk_Storagecurr > Cache.Disk_Storagemax then
41:        Disk_Storagecurr = Runtime_diskcurr + Storagecurr ▷ If there are no free images
42:      if Free'.isEmpty() then
43:        String to_remove = min_timestamp(Running') ▷ Finding service id with min timestamp
44:        Image remove = Running'.get(to_remove)
45:        Free'.put(to_remove, remove) ▷ Adding to Free set of images
46:        CPU'curr -= remove.CPU ▷ Updating current CPU status
47:        RAM'curr -= remove.RAM ▷ Updating current RAM status
48:        Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
49:        Running'.remove(to_remove) ▷ Removing from running set of images
50:      else
51:        String to_remove = min_timestamp(Free') ▷ Finding service with min timestamp
52:        Image remove = Free'.get(to_remove)
53:        Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
54:        Storage'curr -= remove.Storage ▷ Updating current storage status
55:        Free'.remove(to_remove) ▷ Removing the service from cache
56:      end if
57:    end if
58:  end while
59:  ▷ continued...
```

Algorithm LRU Replacement in Edge Cloud - continued

```
60:      ▷ Continued...
61:      Cache.Running  $\leftarrow$  Running'
62:      Cache.Free  $\leftarrow$  Free'
63:      Cache.varcurr = var'curr for var  $\in$  {CPU, RAM, Runtime_disk, Storage}
64:
65:      Cache.Running.put(service_id, new Image(tokens[0],curr_timestamp,tokens[2],tokens[3],tokens[4],tokens[5], storage)
66:                                ▷ Cloning new service with Current Timestamp
67:      if Cache.Free.containsKey(service_id) then                                ▷ If service is also in free list, remove it from free list
68:          Cache.Free.remove(service_id)
69:      end if
70:      Cache.CPUcurr += CPUreq                                ▷ Updating current CPU status
71:      Cache.RAMcurr += RAMreq                                ▷ Updating current RAM status
72:      Cache.Runtime_diskcurr += Runtime_diskreq            ▷ Updating current Runtime Disk status
73:      Cache.Storagecurr += Storagereq                        ▷ Updating current storage in cache
74:      Cache.no_services ++                                ▷ Increment no of services
75:  end if
76: end while
```

Algorithm FIFO replacement in Edge Cloud

```
1: int cache_hit = 0 ▷ No of cache hit
2: while Cloud receives request do
3:   String[ ] tokens = request.split() ▷ Contains all information in the request
4:   String service_id = tokens[0] ▷ Generate service id for the service
5:   double CPUreq = tokens[3], RAMreq = tokens[4], Runtime_Diskreq = tokens[5] ▷ Service requirements
6:   double Storagereq = 0 ▷ Assuming it be 0 for now
7:
8:   if Cache.Running.containsKey(service_id) then ▷ If the service is cached
9:     cache_hit ++
10:  else if Cache.Free.containsKey(service_id) then ▷ If the image of service is present in Free list
11:    Image_replace = Cache.Free.get(service_id)
12:    if There are enough resources in the edge cloud then
13:      replace.timestamp = current_timestamp
14:      Cache.Running.put(service_id, replace) ▷ Adding it to the Running list
15:      Cache.CPUcurr += replace.CPU ▷ Updating current CPU status
16:      Cache.RAMcurr += replace.RAM ▷ Updating current RAM status
17:      Cache.Runtime_diskcurr += replace.Runtime_disk ▷ Updating current Runtime Disk status
18:      Cache.Storagecurr += replace.Storage ▷ Updating current storage status
19:      Cache.Free.remove(replace) ▷ Removing it from free list
20:    end if
21:  else ▷ If the service is not cached
22:    Running' ← Cache.Running
23:    Free' ← Cache.Free ▷ Creating copies. Update cloud only if enough resources are available
24:    CPU'curr ← Cache.CPUcurr
25:    RAM'curr ← Cache.RAMcurr ▷ Creating copies of Cache Status
26:    Runtime_disk'curr ← Cache.Runtime_diskcurr
27:    Storage'curr ← Cache.Storagecurr
28:    while There is no enough resources in edge cloud do
29:      if CPUreq + CPU'curr > Cache.CPUmax || RAMreq + RAM'curr > Cache.RAMmax then
30:        String to_remove = min_timestamp(Running') ▷ Finding service id with least time stamp
31:        Image remove = Running'.get(to_remove)
32:        Free'.put(to_remove, remove) ▷ Inserting its image to set of free images
33:        CPU'curr -= remove.CPU ▷ Updating current CPU status
34:        RAM'curr -= remove.RAM ▷ Updating current RAM status
35:        Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
36:        Running'.remove(to_remove) ▷ Removing its image from set of running images
37:      end if
38:      if Disk_Storagereq + Disk_Storagecurr > Cache.Disk_Storagemax then
39:        Disk_Storagecurr = Runtime_diskcurr + Storagecurr ▷ If there are no free images
40:        if Free'.isEmpty() then
41:          String to_remove = min_timestamp(Running') ▷ Finding service id with min timestamp
42:          Image remove = Running'.get(to_remove)
43:          Free'.put(to_remove, remove) ▷ Adding to Free set of images
44:          CPU'curr -= remove.CPU ▷ Updating current CPU status
45:          RAM'curr -= remove.RAM ▷ Updating current RAM status
46:          Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
47:          Running'.remove(to_remove) ▷ Removing from running set of images
48:        else
49:          String to_remove = min_timestamp(Free') ▷ Finding service with min timestamp
50:          Image remove = Free'.get(to_remove)
51:          Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
52:          Storagecurr -= remove.Storage ▷ Updating current storage status
53:          Free'.remove(to_remove) ▷ Removing the service from cache
54:        end if
55:      end if
56:    end while
57:    ▷ Continued...
```

Algorithm FIFO replacement in Edge Cloud - continued...

```
58:      ▷ Continued...
59:       $Cache.Running \leftarrow Running' \ \& \ Cache.Free \leftarrow Free'$ 
60:       $Cache.var_{curr} = var'_{curr}$  for  $var \in \{CPU, RAM, Runtime\_disk, Storage\}$ 
61:       $Cache.Running.put(service\_id, new Image(tokens[0], curr\_timestamp, tokens[2], tokens[3], tokens[4], tokens[5], storage)$ 
62:                                          ▷ Cloning new service
63:      if  $Cache.Free.containsKey(service\_id)$  then                                ▷ If service is also in free list, remove it from free list
64:           $Cache.Free.remove(service\_id)$ 
65:      end if
66:       $Cache.CPU_{curr} += CPU_{req}$                                           ▷ Updating current CPU status
67:       $Cache.RAM_{curr} += RAM_{req}$                                           ▷ Updating current RAM status
68:       $Cache.Runtime\_disk_{curr} += Runtime\_disk_{req}$                     ▷ Updating current Runtime Disk status
69:       $Cache.Storage_{curr} += Storage_{req}$                                 ▷ Updating current storage in cache
70:       $Cache.no\_services ++$                                               ▷ Increment no of services
71:  end if
72: end while
```

Algorithm LFU replacement in Edge Cloud

```
1: int cache_hit = 0 ▷ No of cache hit
2: while Cloud receives request do
3:   String[ ] tokens = request.split() ▷ Contains all information in the request
4:   String service_id = tokens[0] ▷ Generate service id for the service
5:   double CPUreq = tokens[3], RAMreq = tokens[4], Runtime_Diskreq = tokens[5] ▷ Service requirements
6:   double Storagereq = 0 ▷ Assuming it be 0 for now
7:   if Cache.Running.containsKey(service_id) then ▷ If the service is cached
8:     cache_hit ++
9:     Image_to_update = Cache.Running.get(service_id)
10:    to_update.frequency++ ▷ Increase its frequency by 1
11:    Cache.Running.put(service_id, to_update)
12:  else if Cache.Free.containsKey(service_id) then ▷ If the image of service is present in Free list
13:    Image_replace = Cache.Free.get(service_id)
14:    if There are enough resources in the edge cloud then
15:      replace.frequency = 1
16:      Cache.Running.put(service_id, replace) ▷ Adding it to the Running list
17:      Cache.CPUcurr += replace.CPU ▷ Updating current CPU status
18:      Cache.RAMcurr += replace.RAM ▷ Updating current RAM status
19:      Cache.Runtime_diskcurr += replace.Runtime_disk ▷ Updating current Runtime Disk status
20:      Cache.Storagecurr += replace.Storage ▷ Updating current storage status
21:      Cache.Free.remove(replace) ▷ Removing it from free list
22:    end if
23:  else ▷ If the service is not cached
24:    Running' ← Cache.Running
25:    Free' ← Cache.Free ▷ Creating copies. Update cloud only if enough resources are available
26:    CPU'curr ← Cache.CPUcurr
27:    RAM'curr ← Cache.RAMcurr & ▷ Creating copies of Cache Status
28:    Runtime_disk'curr ← Cache.Runtime_diskcurr
29:    Storage'curr ← Cache.Storagecurr
30:    while There is no enough resources in edge cloud do
31:      if CPUreq + CPU'curr > Cache.CPUmax || RAMreq + RAM'curr > Cache.RAMmax then
32:        String to_remove = min_frequency(Running') ▷ Finding service id with least frequency
33:        Image_remove = Running'.get(to_remove)
34:        Free'.put(to_remove, remove) ▷ Inserting its image to set of free images
35:        CPU'curr -= remove.CPU ▷ Updating current CPU status
36:        RAM'curr -= remove.RAM ▷ Updating current RAM status
37:        Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
38:        Running'.remove(to_remove) ▷ Removing its image from set of running images
39:      end if
40:      if Disk_Storagereq + Disk_Storagecurr > Cache.Disk_Storagemax then
41:        Disk_Storagecurr = Runtime_diskcurr + Storagecurr ▷ If there are no free images
42:        if Free'.isEmpty() then
43:          String to_remove = min_frequency(Running') ▷ Finding service id with least frequency
44:          Image_remove = Running'.get(to_remove)
45:          Free'.put(to_remove, remove) ▷ Adding to Free set of images
46:          CPU'curr -= remove.CPU ▷ Updating current CPU status
47:          RAM'curr -= remove.RAM ▷ Updating current RAM status
48:          Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
49:          Running'.remove(to_remove) ▷ Removing from running set of images
50:        else
51:          String to_remove = min_frequency(Free') ▷ Finding service with least frequency
52:          Image_remove = Free'.get(to_remove)
53:          Runtime_Disk'curr -= remove.Runtime_disk ▷ Updating current runtime disk status
54:          Storage'curr -= remove.Storage ▷ Updating current storage status
55:          Free'.remove(to_remove) ▷ Removing the service from cache
56:        end if
57:      end if
58:    end while
59:    ▷ Continued...
```

Algorithm LFU replacement in Edge Cloud - continued

```
60:      ▷ Continued...
61:       $Cache.Running \leftarrow Running' \ \& \ Cache.Free \leftarrow Free'$ 
62:       $Cache.var_{curr} = var'_{curr}$  for  $var \in \{CPU, RAM, Runtime\_disk, Storage\}$ 
63:       $Cache.Running.put(service\_id, new\ Image(tokens[0],1,tokens[2],tokens[3],tokens[4],tokens[5], storage)$ 
64:                                          ▷ Cloning new service with frequency 1
65:      if  $Cache.Free.containsKey(service\_id)$  then                                ▷ If service is also in free list, remove it from free list
66:           $Cache.Free.remove(service\_id)$ 
67:      end if
68:       $Cache.CPU_{curr} += CPU_{req}$                                           ▷ Updating current CPU status
69:       $Cache.RAM_{curr} += RAM_{req}$                                           ▷ Updating current RAM status
70:       $Cache.Runtime\_disk_{curr} += Runtime\_disk_{req}$                     ▷ Updating current Runtime Disk status
71:       $Cache.Storage_{curr} += Storage_{req}$                                 ▷ Updating current storage in cache
72:       $Cache.no\_services ++$                                               ▷ Increment no of services
73:  end if
74: end while
```

Algorithm GDSF replacement in Edge Cloud

```
1: int cache_hit = 0 ▷ No of cache hit
2: double clock = 0.00 ▷ Clock for GDSF priority calculation
3: while Cloud receives request do
4:   String[ ] tokens = request.split() ▷ Contains all information in the request
5:   String service_id = tokens[0] ▷ Generate service id for the service
6:   double CPU_req = tokens[3], RAM_req = tokens[4], Runtime_Disk_req = tokens[5] ▷ Service requirements
7:   double Storage_req = 0 ▷ Assuming it be 0 for now
8:   if Cache.Running.containsKey(service_id) then ▷ If the service is cached
9:     cache_hit ++
10:    Image_to_update = Cache.Running.get(service_id)
11:    to_update.frequency++ ▷ Increase its frequency by 1
12:    to_update.priority = priority_calc(clock, to_update.frequency, Storage_req)
13: ▷ Recalculating priority with updated frequency
14:    Cache.Running.put(service_id, to_update)
15: else if Cache.Free.containsKey(service_id) then ▷ If the image of service is present in Free list
16:   Image_replace = Cache.Free.get(service_id)
17:   if There are enough resources in the edge cloud then
18:     replace.frequency = 1
19:     replace.priority = priority_calc(clock, replace.frequency, Storage_req)
20:     Cache.Running.put(service_id, replace) ▷ Adding it to the Running list
21:     Cache.CPU_curr += replace.CPU ▷ Updating current CPU status
22:     Cache.RAM_curr += replace.RAM ▷ Updating current RAM status
23:     Cache.Runtime_disk_curr += replace.Runtime_disk ▷ Updating current Runtime Disk status
24:     Cache.Storage_curr += replace.Storage ▷ Updating current storage status
25:     Cache.Free.remove(replace) ▷ Removing it from free list
26:   end if
27: else ▷ If the service is not cached
28:   Running' ← Cache.Running
29:   Free' ← Cache.Free ▷ Creating copies. Update cloud only if enough resources are available
30:   CPU'_curr ← Cache.CPU_curr
31:   RAM'_curr ← Cache.RAM_curr ▷ Creating copies of Cache Status
32:   Runtime_disk'_curr ← Cache.Runtime_disk_curr
33:   Storage'_curr ← Cache.Storage_curr
34:   clock' ← clock
35:   while There is no enough resources in edge cloud do
36:     if CPU_req + CPU'_curr > Cache.CPU_max || RAM_req + RAM'_curr > Cache.RAM_max then
37:       String to_remove = min_priority(Running') ▷ Finding service id with least priority
38:       Image_remove = Running'.get(to_remove)
39:       Free'.put(to_remove, remove) ▷ Inserting its image to set of free images
40:       CPU'_curr -= remove.CPU ▷ Updating current CPU status
41:       RAM'_curr -= remove.RAM ▷ Updating current RAM status
42:       Runtime_Disk'_curr -= remove.Runtime_disk ▷ Updating current runtime disk status
43:       clock' += remove.priority ▷ Updating clock
44:       Running'.remove(to_remove) ▷ Removing its image from set of running images
45:     end if
46:     if Disk_Storage_req + Disk_Storage_curr > Cache.Disk_Storage_max then
47:       ▷ Disk_Storage_curr = Runtime_disk_curr + Storage_curr
48:       if Free'.isEmpty() then ▷ If there are no free images
49:         String to_remove = min_priority(Running') ▷ Finding service id with least priority
50:         Image_remove = Running'.get(to_remove)
51:         Free'.put(to_remove, remove) ▷ Adding to Free set of images
52:         CPU'_curr -= remove.CPU ▷ Updating current CPU status
53:         RAM'_curr -= remove.RAM ▷ Updating current RAM status
54:         Runtime_Disk'_curr -= remove.Runtime_disk ▷ Updating current runtime disk status
55:         clock' += remove.priority ▷ Updating clock
56:         Running'.remove(to_remove) ▷ Removing from running set of images
57:       else
58:         String to_remove = min_priority(Free') ▷ Finding service with least priority
59:         Image_remove = Free'.get(to_remove)
60:         ▷ Continued...
```

Algorithm GDSF replacement in Edge Cloud - continued

```
61:           ▷ Continued...
62:            $Runtime\_Disk'_{curr} \leftarrow remove.Runtime\_disk$            ▷ Updating current runtime disk status
63:            $Storage'_{curr} \leftarrow remove.Storage$            ▷ Updating current storage status
64:            $Free'.remove(to\_remove)$            ▷ Removing the service from cache
65:       end if
66:   end if
67: end while
68:  $Cache.Running \leftarrow Running' \ \& \ Cache.Free \leftarrow Free'$ 
69:  $Cache.var_{curr} = var'_{curr}$  for  $var \in \{CPU, RAM, Runtime\_disk, Storage\}$ 
70:
71: double Pr = priority_calc(clock, 1, Storage_req)           ▷ Calculating priority of new service
72:  $Cache.Running.put(service\_id, new Image(tokens[0], 1, Pr, tokens[2], tokens[3], tokens[4], tokens[5], storage))$ 
73:           ▷ Cloning new service with frequency 1
74: if  $Cache.Free.containsKey(service\_id)$  then           ▷ If service is also in free list, remove it from free list
75:      $Cache.Free.remove(service\_id)$ 
76: end if
77:  $Cache.CPU_{curr} += CPU_{req}$            ▷ Updating current CPU status
78:  $Cache.RAM_{curr} += RAM_{req}$            ▷ Updating current RAM status
79:  $Cache.Runtime\_disk_{curr} += Runtime\_disk_{req}$            ▷ Updating current Runtime Disk status
80:  $Cache.Storage_{curr} += Storage_{req}$            ▷ Updating current storage in cache
81:  $Cache.no\_services ++$            ▷ Increment no of services
82: end if
83: end while
```

Algorithm SCRP score calculator

```
1: ▷ Function to calculate SCRP score of a service for S-cache replacement
2: double SCRP_calc(Image service, Cache cache)
3:   double Disk = service.Runtime_disk + service.Storage
4:   double R =  $\frac{cache.CPU_{curr}}{cache.CPU_{max}} \times service.CPU + \frac{cache.RAM_{curr}}{cache.RAM_{max}} \times service.RAM + \frac{cache.Disk_{curr}}{cache.Disk_{max}} \times service.Disk$ 
5:
6:   if R == 0 then                                     ▷ If no resources are being used
7:     return Double.MAX_VALUE
8:   end if
9:
10:  return  $\frac{service.frequency}{service.stay \times R}$                                      ▷ SCRP score of the service
```

Algorithm Least SCRP score Finder

```
1: ▷ Function to find service id that has least SCRP score from the set of images for SCRP
2: String min_SCRP(HashMap< String, Image >hm, Cache cache)
3:   String min_SCRP = ""
4:   double SCRP_min = Double.MAX_VALUE
5:   hm.forEach((Service_id, Image) →                                     ▷ Iterating over the set of images
6:     Image.SCRP = SCRP_calc(Image, cache)                             ▷ Calculating the SCRP score for the service
7:     if Image.SCRP < SCRP_min then
8:       SCRP_min = Image.SCRP
9:       min_SCRP = Service_id
10:    end if
11:  return min_SCRP                                     ▷ Service id of the service having least SCRP score
```

Algorithm Least Priority Finder for GDSF in SCRP

```
1: ▷ Function to find service id that has least priority from the set of images
2: String min_priority(double clock, HashMap< String, Image > hm, Cache cache)
3:   String min_priority = ""
4:   double priority_min = Double.MAX_VALUE
5:   hm.forEach((Service_id, Image) →                                     ▷ Iterating over whole set of images
6:     double pr = cache.priority_calc(clock, Image.stay, Image.Storage)   ▷ Calculating the priority for each image
7:     if pr < priority_min then
8:       priority_min = Image.priority
9:       min_priority = Service_id
10:    end if
11:
12:  return min_priority                                     ▷ Service id having least priority
```

Algorithm SCRP replacement in Edge Cloud

```
1: int cache_hit = 0
2: double clock = 0.00                                ▷ Clock for GDSF replacement
3: while Cloud receives request do
4:   Cache.Running.forEach((Service_id, Image) →        ▷ Increase stay of each service in Running list
5:     Image.stay++
6:   Cache.Free.forEach((Service_id, Image) →           ▷ Increase stay of each service in Free list
7:     Image.stay++
8:   String[ ] tokens = request.split()                ▷ Contains all information in the request
9:   String service_id = tokens[0]                    ▷ Generate service id for the service
10:  double CPUreq = tokens[3], RAMreq = tokens[4], Runtime_Diskreq = tokens[5]    ▷ Service requirements
11:  double Storagereq = 0                             ▷ Assuming it be 0 for now
12:  Image New_service = new Image(service_id,1,1,0,tokens[2],tokens[3],tokens[4],tokens[5], Storagereq)    ▷ New service image
13:  New_service.SCRP = SCRP_calc(New_service, cache)    ▷ Calculating SCRP score for new service
14:  if Cache.Running.containsKey(service_id) then      ▷ If the service is cached
15:    Image to_update = Cache.Running.get(service_id)
16:    to_update.frequency++                            ▷ Increase its frequency by 1
17:    Cache.Running.put(service_id, to_update)
18:    cache_hit++
19:  else if Cache.Free.containsKey(service_id) then    ▷ If the image of service is present in Free list
20:    Image replace = Cache.Free.get(service_id)
21:    replace.frequency++                              ▷ Increase its frequency by 1
22:    if There are enough resources in the edge cloud then
23:      Cache.Running.put(service_id, replace)        ▷ Adding it to the Running list
24:      Cache.CPUcurr += replace.CPU                 ▷ Updating current CPU status
25:      Cache.RAMcurr += replace.RAM                 ▷ Updating current RAM status
26:      Cache.Runtime_diskcurr += replace.Runtime_disk    ▷ Updating current Runtime Disk status
27:      Cache.Storagecurr += replace.Storage         ▷ Updating current storage status
28:      Cache.Free.remove(replace)                   ▷ Removing it from free list
29:    else
30:      Cache.Free.put(service_id, replace)
31:    end if
32:  else                                              ▷ If the service is not cached
33:    Running' ← Cache.Running
34:    Free' ← Cache.Free                             ▷ Creating copies. Update cloud only if enough resources are available
35:    CPU'curr ← Cache.CPUcurr
36:    RAM'curr ← Cache.RAMcurr                        ▷ Creating copies of Cache Status
37:    Runtime_disk'curr ← Cache.Runtime_diskcurr
38:    Storage'curr ← Cache.Storagecurr
39:    while There is no enough resources in edge cloud do
40:      if CPUreq + CPU'curr > Cache.CPUmax || RAMreq + RAM'curr > Cache.RAMmax then
41:        String to_remove = min_SCRP(Running', cache)    ▷ Finding service id with least priority
42:        Image remove = Running'.get(to_remove)
43:        if New_service.score > remove.score then
44:          Free'.put(to_remove, remove)                ▷ Inserting its image to set of free images
45:          CPU'curr -= remove.CPU                    ▷ Updating current CPU status
46:          RAM'curr -= remove.RAM                    ▷ Updating current RAM status
47:          Runtime_Disk'curr -= remove.Runtime_disk    ▷ Updating current runtime disk status
48:          Running'.remove(to_remove)                ▷ Removing its image from set of running images
49:        else
50:          break
51:        end if
52:      end if
53:      if Disk_Storagereq + Disk_Storagecurr > Cache.Disk_Storagemax then
54:        Disk_Storagecurr = Runtime_diskcurr + Storagecurr    ▷ If there are no free images
55:        if Free'.isEmpty() then
56:          String to_remove = min_SCRP(Running', cache)    ▷ Finding service id with least frequency
57:          Image remove = Running'.get(to_remove)
58:          if New_service.score > remove.score then
59:            Free'.put(to_remove, remove)                ▷ Adding to Free set of images
60:            CPU'curr -= remove.CPU                    ▷ Updating current CPU status
61:            RAM'curr -= remove.RAM                    ▷ Updating current RAM status
62:          end if
        end if
        ▷ Continued...
```

Algorithm SCRP replacement in Edge Cloud - continued

```
63:           ▷ Continued...
64:           Runtime_Disk'curr -= remove.Runtime_disk           ▷ Updating current runtime disk status
65:           Running'.remove(to_remove)           ▷ Removing from running set of images
66:       else
67:           break
68:       end if
69:   else
70:       String to_remove = min_Priority(clock, Free', cache)           ▷ Finding service with least priority based on GDSF
71:       Image remove = Free'.get(to_remove)
72:       Runtime_Disk'curr -= remove.Runtime_disk           ▷ Updating current runtime disk status
73:       Storage'curr -= remove.Storage           ▷ Updating current storage status
74:       clock += priority_calc(clock, remove.stay, remove.storage)
75:       Free'.remove(to_remove)           ▷ Removing the service from cache
76:   end if
77: end if
78: end while
79: Cache.Running ← Running' & Cache.Free ← Free'
80: Cache.varcurr = var'curr for var ∈ {CPU, RAM, Runtime_disk, Storage}
81: Cache.Running.put(service_id, New_service)           ▷ Cloning new service with frequency 1
82: if Cache.Free.containsKey(service_id) then           ▷ If service is also in free list, remove it from free list
83:     Cache.Free.remove(service_id)
84: end if
85: Cache.CPUcurr += CPUreq           ▷ Updating current CPU status
86: Cache.RAMcurr += RAMreq           ▷ Updating current RAM status
87: Cache.Runtime_diskcurr += Runtime_diskreq           ▷ Updating current Runtime Disk status
88: Cache.Storagecurr += Storagereq           ▷ Updating current storage in cache
89: Cache.no_services ++           ▷ Increment no of services
90: end if
91: end while
```
