
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issues 4, pp. 649–660, DOI 10.12694:/scpe.v21i4.1799

SCIENTIFIC APPLICATIONS IN THE CLOUD: RESOURCE OPTIMISATION BASED ON METAHEURISTICS

ANAS MOKHTARI*, MOSTAFA AZIZI† AND MOHAMMED GABLI‡

Abstract. The advent of emerging technologies such as 5G and Internet of Things (IoT) will generate a colossal amount of data that should be processed by the cloud computing. Thereby, cloud resources optimisation represents significant benefits in different levels: cost reduction for the user, saving energy consumed by cloud data centres, etc. Cloud resource optimisation is a very complex task due to its NP-hard characteristic. In this case, use of metaheuristic approaches is more rational. But the quality of metaheuristic solutions changes by changing the problem. In this paper we have dealt with the problem of determining the configuration of resources in order to minimise the payment cost and the duration of the scientific applications execution. For that, we proposed a mathematical model and three metaheuristic approaches, namely the Genetic Algorithm (GA), hybridisation of the Genetic Algorithm with Local Search (GA-LS) and the Simulated Annealing (SA). The comparison between them showed that the simulated annealing finds more optimal solutions than those proposed by the genetic algorithm and the GA-LS hybridisation.

Key words: Cloud computing, Resources Management, Optimisation, Metaheuristic, Artificial Intelligence

AMS subject classifications. 68M14, 68T01, 90C10

1. Introduction. The cloud computing is a distributed computer system based on an emergent technologies like the virtualisation. By comparing it with the conventional distributed computing, the latter has the objective of providing a collaborative sharing of resources to which users are linked, while the cloud computing has the objective of providing services or applications with ensuring of the scaling, the transparency (vis-a-vis the physical implementation of the cloud), the security, the supervision and the management.

There are three main types of cloud models:

IaaS (Infrastructure as a Service) offers a low level computing resources in the form of Virtual Machines (VM). EC2 [1] and Azure [3] are two IaaS examples provided respectively by Amazon and Microsoft.

PaaS (Platform as a Service) offers ready to use software platforms on which users develop and deploy their applications. Heroku [6] and Google App Engine [4] provide this type of cloud.

SaaS (Service as a Service) offers ready-to-use applications. It's the highest level in the cloud. G Suite [5] (Google) and Office 365 [7] (Microsoft) are SaaS models.

There are numerous applications that require High Performance Computing (HPC). Weather forecast, chemical process modeling, and the physics simulations are examples of such applications. Since the available computing resources in the cloud are very efficient, users of the HPC applications showed interest in running their intensive applications in the cloud environment.

Cloud computing face several challenges and problems that are subject to scientific research. The excessive consumption of the electric energy and the performance degradation of the application execution because of underestimation of reserved resources are examples of problems that cloud systems face.

This work addresses the problem of performance and cost of applications execution in the cloud. It's a double contradictory objectives problem. The first one is to maximise the resources to be used to have a good performance by reducing the execution time. The second one is to minimise the resources to be used to reduce the payment cost.

There are several works in the literature which have dealt with this type of problem. The complexity of the cloud resource optimisation problem has led researchers to use metaheuristics. For instance, in our

*MATSI Lab., ESTO, University Mohammed Ist, Oujda, Morocco (a.mokhtari@ump.ac.ma).
†MATSI Lab., ESTO, University Mohammed Ist, Oujda, Morocco (azizi.mos@ump.ac.ma).
‡LARI Lab., FSO, University Mohammed Ist, Oujda, Morocco (medgabli@ump.ac.ma).

TABLE 2.1
Definition of variables

Variable	Definition
P	Set of packages types offered by a cloud provider during a set of time periods
C_M	Set of consumer requirements as the maximum cost
T_M	Maximum time for execution
D_S	Disk storage
M_C	Memory capacity
G_f	Processing demand in Gflop
p	A package type from the set P ($p \in P$)
c_p	The cost of purchasing the package p for one period of time
d_p	Disk storage computing resource for the package p
m_p	Memory capacity for the package p
g_p	Processing power for the package p
N_M	Maximum limit of packages that a consumer can purchase at a period of time

previous works [20, 21, 22], we considered weighted objective functions in our model and chose the approach of genetic algorithms for dealing with this issue. The authors of [25] investigated the meta-heuristic resource allocation techniques used in the IaaS cloud computing environment. Jena et al. [26] proposed a hybridisation of modified Particle swarm optimization (MPSO) and improved Q-learning algorithm to load balancing of tasks on the cloud environment. Coutinho et al. [10] implemented an ILP model and then used Greedy Randomised Adaptive Search Procedure (GRASP) to solve the resource optimisation problem. Except that the proposed metaheuristics has been compared with a deterministic algorithm. Due to difficulty of this problem, which is classified as NP-hard [10], most of these works have tried to solve it using metaheuristic approaches. The effectiveness of these approaches in terms of the solutions found varies from one problem to another.

In this work, we used three metaheuristic algorithms: Genetic Algorithm (GA), Simulated Annealing (SA) and hybridisation of GA and Local Search method (GA-LS). Our goal is executing demanding HPC applications in the IaaS cloud.

The remaining of this paper is organised as follows: in the *Sect.* 2, we describe the mathematical model used. In the *Sect.* 3, we explain the three proposed resolution metaheuristics. In the *Sect.* 4, We compare between these approaches by experiments, then analyse the obtained results. We conclude in *Sect.* 5.

2. Mathematical model. In the infrastructure cloud (IaaS), resources are the virtual CPUs (vCPU), the memory, the storage drives, etc. of the VMs. Each resource type has the same characteristics as the equivalent physical resources (like processor frequency for the vCPU).

Cloud infrastructure service providers make available to the users a variety of VM choices. Each type has a specific computing power and usage price per hour.

The mathematical model that we present is based on the variables cited in the *Table* 2.1.

Our optimisation problem is composed of two objective functions: (i) minimise the total cost and (ii) minimise the execution time. We transformed them to a unique objective function by their sum with coefficients assigned to each of these two functions (weighted sum). So, we have [10]:

$$\text{(CC ILP)} \qquad min(\alpha f_1 + (1-\alpha) f_2) \tag{2.1}$$

with

$$f_1 = \sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} c_p x_{pit} \tag{2.2}$$

and

$$f_2 = t_m \tag{2.3}$$

subject to

$$\sum_{p\in P}\sum_{i=1}^{N_M}\sum_{t\in T} c_p x_{pit} \leq C_M \tag{2.4}$$

$$\sum_{p\in P}\sum_{i=1}^{N_M} d_p x_{pit} \geq D_s x_{p'i't}, \qquad \forall t \in T, \forall p' \in P, \forall i' \in \{1, ..., N_M\} \tag{2.5}$$

$$\sum_{p\in P}\sum_{i=1}^{N_M} m_p x_{pit} \geq M_C x_{p'i't}, \qquad \forall t \in T, \forall p' \in P, \forall i' \in \{1, ..., N_M\} \tag{2.6}$$

$$\sum_{p\in P}\sum_{i=1}^{N_M}\sum_{t\in T} g_p x_{pit} \geq G_f \tag{2.7}$$

$$\sum_{p\in P}\sum_{i=1}^{N_M} x_{pit} \leq N_M, \qquad \forall t \in T \tag{2.8}$$

$$t_m \geq t x_{pit}, \qquad \forall t \in T, \forall p \in P, \forall i \in \{1, ..., N_M\} \tag{2.9}$$

$$x_{pit+1} \leq x_{pit}, \qquad \forall t \in T, \forall p \in P, \forall i \in \{1, ..., N_M\} \tag{2.10}$$

$$x_{pi+1t} \leq x_{pit}, \qquad \forall t \in T, \forall p \in P, \forall i \in \{1, ..., N_M - 1\} \tag{2.11}$$

$$x_{pit} \in \{0, 1\}, \qquad \forall t \in T, \forall p \in P, \forall i \in \{1, ..., N_M\} \tag{2.12}$$

$$t_m \in \mathbb{Z} \tag{2.13}$$

The terms α and $(1 - \alpha)$ represent the weights of the two objectives. To automate the choice of the weights and ensure a fair treatment between the two objective functions [12], we used dynamic weights to meet the condition (2.14):

$$|\alpha(t) f_1 - (1 - \alpha(t)) f_2| \prec \varepsilon, \tag{2.14}$$

where ε is a positive number in the vicinity of 0, t is a time-step, and $\alpha(t)$ and $(1 - \alpha(t))$ are the dynamic weights. $\alpha(t)$ is calculated by the formula (2.15):

$$\alpha(t) = \frac{|f_2(x_{t-1})|}{|f_1(x_{t-1})| + |f_2(x_{t-1})|} \tag{2.15}$$

where x_{t-1} is the best solution of the iteration $(t - 1)$ of the metaheuristic. For more details, see our previous work [20].

TABLE 3.1
The SA variables definitions.

Variable	Definition
S	Solution that represents a combination of packages.
Δ	The difference between the value of the optimisation function of the current solution and that under the evaluation.
T	System temperature.

3. Metaheuristic resolution approaches.

3.1. Simulated Annealing. The origin of this method come from the experiments done by Metropolis et al. [19] to simulate the stochastic evolution of such physical system. In the context of minimisation of the objective function f, the process of simulated annealing is illustrated in the Algorithm 5. *Table* 3.1 defines the used variables in this algorithm.

Algorithm 1 Simulated Annealing

```
1: Generate an initial solution S ← S_0
2: Initiate the temperature T ← T_0
3: repeat
4:     for a predetermined number of iterations do
5:         Generate a solution S_0 in the vicinity of S
6:         Calculate Δ = f(S_0) − f(S)
7:         if Δ < 0 then
8:             S ← S_0
9:         else
10:            S ← S_0 with the probability e^(−Δ/T)
11:        end if
12:    end for
13:    Decrease the temperature T
14: until The stop criterion is satisfactory
15: Return to the best configuration found
```

The proper functioning of the Simulated Annealing algorithm depends on the configuration space and the temperature decrease function. For example, Dréo et al. [11] and Kirkpatrick et al. [18] propose methods to define this function.

3.2. Genetic Algorithm. The Genetic Algorithm is an optimisation technique that mimic the natural evolution. The first work on the GA was developed by John Holland in 1962 [16]. The popularity of the GA returns to the work of David Golberg [14]. In this context, we call *chromosome* x any proposed solution by the GA. This chromosome is composed of a set of values called *genes*. The Algorithm 6 presents the second approach. *Table* 3.2 describes used variables.

The fitness function permits to evaluate each chromosome by assigning it a value that depends on the quality of this solution (chromosome). The crossover operation randomly chooses two chromosomes and a position of crossing. Then, we exchange the bits which are very close to the position of crossing. The mutation operation is to select a gene and replace its value by another randomly chosen from a given set.

3.3. Hybridisation of GA and Local Search method. The metaheuristic methods of optimisation are classified into two categories: approaches which aim at diversification and those which aim at intensification in the research space. The main difference between these two categories [13] is that in intensification, research focuses on the evaluation of neighbours of elite solutions, while diversification encourages the research process to evaluate regions not visited and to generate solutions that differ significantly from the solutions seen previously.

Algorithm 2 Genetic Algorithm

1: Initialise $t \leftarrow 0$. Randomly generate an initial population of individuals $P(0)$
2: Put $f(x,t) = \alpha(t)f_1(x) + (1-\alpha(t))f_2(x)$ and $\alpha(0) = 0.5$
3: **for** A predetermined number of iterations **do**
4: Put $t \leftarrow t+1$. Apply on each solution (chromosome) of the current population an evaluation by the fitness function f
5: Apply the selection operator on the current population $P(t)$ to produce a new population $P'(t)$
6: Apply the crossover operator on $P'(t)$ with a probability P_c. A new population $P''(t)$ is created
7: Apply the mutation operator on $P''(t)$ with a probability P_m. The result population is noted $P(t+1)$
8: Update the weight $\alpha(t)$ by using the dynamic method described by the formula (2.15)
9: **end for**

TABLE 3.2
Definitions of the genetic algorithm variables

Variable	Definition
$P(t)$	Population composed of several candidate solutions used in the iteration t.
x	A solution that represents a combination of packages.
P_c	Crossover probability.
P_m	Mutation probability.

GA are based on diversification, while the Local Search (LS) algorithm intensifies its search in the vicinity of a solution.

Evolutionary algorithms, including GA, suffer from the inability to intensify research enough. As a result, they cannot effectively achieve high quality candidate solutions [17]. A significant improvement in the performance of the GA for combinatorial problems can be ensured by the application of LS on the solution found by the GA. We talk about *hybridisation between GA and LS* (GA-LS).

The third approach is presented by the Algorithm 7. We are considering the improvement of the solution proposed by the GA using LS as a method of *intensification* [8]. The first part of this algorithm (lines 1 to 10) represents the GA. It consists in initialising a population of candidate solutions, then repetitively carrying out the operations of evaluation, selection, crossing and mutation. In the second part (lines 11 to 18), which represents the LS algorithm, the best solution found by the GA, noted s, is selected to become the initial configuration of LS.

4. Comparison of the Results.

4.1. Description of data. One of the most important parts of a comparison between metaheuristics is the testbed on which it is made [23]. It is preferable to develop a new test bed (program) to implement the metaheuristics and compare between them. In addition, so that the execution time of each algorithm is comparable to the others, the best approach to use consists in implementing these algorithms by the same programming language, compiling them in the same machine with the same parameters (flags) of compilation and run them in the same machine [23].

We implemented our approaches in *Java* language, then compiled the code by the *javac* compiler (OpenJDK), version 11.0.6, on a machine characterised by an Intel processor ® Core™ i7-2670QM which contains eight cores clocked at 2.20 GHz. We tested our programs on the same computer using the Java virtual machine (OpenJDK JRE), version 11.0.6.

We have applied these algorithms on four instances of applications with different sizes: an application which deals with the problem of manipulation of the biological sequence (*raxml*) [24], a typical analysis application for the CMS experience (*CMS-1500*) [9] and two applications for solving the QAP[1] problem by the separation and

[1]Quadratic Assignment Problem

Algorithm 3 Hybridisation algorithm between the genetic algorithm and the local search (GA-LS)

Set $t \leftarrow 0$
Randomly generate an initial population of individuals $P(0)$
Put $f(x,t) = \alpha(t)f_1(x) + (1-\alpha(t))f_2(x)$ and $\alpha(0) = 0.5$
for A predetermined number of iterations **do**
    Put $t \leftarrow t+1$. Apply on each solution (chromosome) of the current population an evaluation by the fitness function f
    Apply the selection operator on the current population $P(t)$ To produce a new population $P'(t)$
    Apply the crossover operator on $P'(t)$ with a probability P_c. A new population $P''(t)$ is created
    Apply the mutation operator on $P''(t)$ with a probability P_m. The result population is noted $P(t+1)$
    Update the weight $\alpha(t)$ by using the dynamic method described by the formula (2.15)
end for
Choose the solution $s \in S$ found by the genetic algorithm
repeat
    Choose s' in the vicinity of s ($s' \in V(s)$)
    $\Delta = f(s) - f(s')$
    if $\Delta > 0$ **then**
        $s \leftarrow s'$
    end if
until The stop criterion is satisfactory

TABLE 4.1
Characteristics of the applications used in the experiment [10]

Application	Memory (GB)	Storage (GB)	GFLOP	Time (hour)	Max packages	Cost ($)
raxml	3	2	3,317,760	10	20	50
cms-1500	2,250	30	324,000,000	10	20	75
nug28-cbb	528	528	541,765,325	10	20	120
nug30-cbb	918	918	967,438,080	15	30	250

evaluation algorithm (*nug28-cbb* and *nug30-cbb*) [15]. The characteristics of these applications are summarised in the *Table* 4.1.

In order to have realistic results, we applied our simulation to the offers of two cloud computing providers: Amazon EC2 [1] and Google Compute Engine [2] (*cf. Table* 4.2).

For each application, we performed the metaheuristics several times to ensure the accuracy of the results obtained. Details are presented in *Table* 4.3.

To give metaheuristic approaches more chance of finding good solutions, we have extended the execution time for larger application problems (between 1 and 20 seconds, *cf. Table* 4.3-left). In addition, in order to verify the influence of this duration on the quality of the results obtained, we launched other tests with a long execution time (*cf. Table* 4.3-right).

Since the LS depends on the solution found after the execution of the GA, and in order to be able to compare between the results of the GA and GA-LS hybridisation approaches, we have allocated to the LS part an execution time equivalent to approximately 8% of the execution time of the GA alone:

$$\text{Duration(GA-LS)} \approx 1.08 \times \text{duration(GA)} \Rightarrow \text{Duration(GA-LS)} \approx \text{duration(GA)}$$

Generally, the parameters values of each algorithm must be determined by its designer because their change influences the performance of metaheuristics [23]. For the case of our approaches, we fixed the parameters

TABLE 4.2
Cloud Provider's VM Instance Pricing

Cloud provider	Instance type	vCPU	Memory (GB)	Storage (GB)	Price (/hour)
Amazon EC2	c4.large	2	3.75	200	$0.128
	c4.xlarge	4	7.5	400	$0.255
	c4.2xlarge	8	15	800	$0.509
	c4.4xlarge	16	30	1,600	$1.018
	c4.8xlarge	36	60	2,500	$1.938
Google Compute Engine	n1-highcpu-2	2	1.80	100	$0.0764
	n1-highcpu-4	4	3.60	200	$0.1529
	n1-highcpu-8	8	7.20	400	$0.3058
	n1-highcpu-16	16	14.40	800	$0.6116
	n1-highcpu-32	32	28.80	1,600	$1.2233
	n1-highcpu-64	64	57.60	2,500	$2.4077

TABLE 4.3
Parameters used for the execution of the three approaches: GA, GA-LS hybridisation, and SA: for a no long duration (left) and for a long duration (right)

Application	Execution time (s)	Number of trials
raxml	1	10
cms-1500	5	10
nug28-cbb	10	10
nug30-cbb	20	10

Application	Execution time (s)	Number of trials
raxml	180	2
cms-1500	180	2
nug28-cbb	180	2
nug30-cbb	180	2

according to the application concerned. The details are presented in *Table* 4.4 for GA and GA-LS, and *Table* 4.5 for SA.

4.2. Results analysis. One of the most important points in the comparison between metaheuristics is the *solution quality*. The measurement of this quality is relative and it depends on the treated application. For long-term planning, the acceptable difference between the found and the optimal solutions is smaller than that of short-term planning applications [23]. For that, we must compare between the solutions found by metaheuristics and not determine if they reach a threshold of solution quality [23].

To compare the solutions found by metaheuristics, we must base ourselves on a metric. In our case, we have the possibility of using one of the following two metrics [23]:

- Deviation from best-known solutions for a problem;
- Deviation between the algorithms being compared: This method has the advantage of making the comparison between the algorithms very explicit. However, these comparisons lack any sense of the actual error of solutions.

In our knowledge, there is no best-known solutions for the case of our problem. So, we used the second metric.

The results of our experiment are detailed in the *Table* 4.6. We have limited ourselves in this table to the best solution found in the ten trials carried out by application.

In the *Application* column, we put the name of the problem instance to be executed with the name of the cloud on which this execution is planned. For example, the first line (*raxml_am*) concerns the execution of raxml instance in the Amazon EC2 (am) cloud, while for the fifth line (*raxml_go*), we are talking about the same application in the Google Compute Engine (go) cloud.

The details of the solution for each metaheuristic are represented in five columns: the payment cost f_1

TABLE 4.4
Parameters used for the GA

Problem instance	**Population size**	**Crossover probability (P_c)**	**Mutation probability (P_m)**
raxml	40	0.5	0.01
cms-1500	40	0.5	0.01
nug28-cbb	40	0.5	0.01
nug30-cbb	60	0.5	0.01

TABLE 4.5
Parameters used for the SA

Problem instance	**Annealing rate (λ)**	**Initial temperature**
raxml	0.9	100
cms-1500	0.9	100
nug28-cbb	0.9	100
nug30-cbb	0.9	100

(in dollars) of the found solution, the duration f_2 (in hours) necessary for the execution of the application, the value of the weighted objective function f, f_{avg} to check the robustness of this solution by calculating the average of the values of f for the ten trials carried out ($f_{avg} = \sum_{i=1}^{10} \frac{f_{trial_i}}{10}$), and the fifth column contains the execution time of the metaheuristic (in seconds) which allowed to have this result.

From *Fig.* 4.1, we can easily notice that, for all the tested applications, the best minimisation of the objective function f is ensured by the SA, followed by the GA-LS hybridisation algorithm. The optimal cost of f found by the SA is 53.34% (nug30-cbb_go) to 92.3% (raxml_go) less than that found by the GA, and 29.27% (nug28-cbb_go) to 90.57% (raxml_go) less than the GA-LS.

The fact that the GA-LS optimises the objective function better than the GA is expected since the GA-LS diverts the weakness of the GA in terms of intensification.

Since the function f is the result of the weighted sum of the two functions f_1 and f_2 ($f = \alpha f_1 + (1-\alpha) f_2$), we also analysed and compared these two objectives for the three metaheuristic approaches. The histograms in *Fig.* 4.2 and *Fig.* 4.3 represent, respectively, the values of f_1 and f_2.

Simulated Annealing gives us, for all the tested applications, cheaper solutions (f_1) and faster execution (f_2) than those proposed by the GA and the GA-LS. Solutions of the GA-LS hybridisation algorithm are more expensive than those found by the GA (case of nug28-cbb_go and nug30-cbb_go in *Fig.* 4.2), but are always faster in execution (*cf. Fig.* 4.3).

Since the aforementioned results represent the best solutions found among the ten trials that we carried out, this brings us back to verifying that these solutions do not represent particular cases (*noise*). The robustness of the GA is presented in *Fig.* 4.4. The red (dark) bar represents the value f and the red+blue (dark+light) bar represents f_{avg}. The best values of f are 20.5% to 46.35% smaller than the average f_{avg} of the solutions found by the ten trials. Similarly, f of the GA-LS (*cf. Fig.* 4.5) is 20.6% to 49.79% lower than f_{avg}. This variation is more reduced in the case of the SA (*cf Fig.* 4.6) since it is between 0.01% and 29.72%. The long-term execution of the three algorithms (180 seconds) did not considerably improve the quality of the found solutions (details in the *Table* 4.6). In this case too, the SA gives in all the tested cases more optimal solutions than those of the GA and the GA-LS.

5. Conclusion. In this paper, we treated the problem of multi-objective optimisation of the cloud computing resources, for the execution of the intensive computing applications. We took into account the decrease in the budget by reducing the allocation price of computing resources (first objective) and the increase in performance by minimising the execution time (second objective). Then, we implemented three problem solving

TABLE 4.6

Results obtained by the GA, the GA-LS hybridisation and the SA. The first half of the results (8 lines) represent the solutions found by our algorithms executed in durations between 1 and 20 seconds. The second half represents the tests of these algorithms in long durations (180 seconds). The values of f_1, f_2 and f represent the best solution found among the 10 tests launched. The columns of f_1 represent the estimated cost of payment in dollars. The columns of f_2 are the execution times of the application in question, estimated in hours. f is the fitness function (objective function) of the solution found. To ensure the robustness of these solutions, we added the columns of f_{moy} which represent the average of the values of the objective functions of the 10 tests carried out ($f_{moy} = \sum_{i=1}^{10} \frac{f_{test_i}}{10}$).

Application	**Genetic Algorithm**					**GA-LS Hybridisation**					**Simulated Annealing**				
	f_1	f_2	f	f_{avg}	Duration of execution (s)	f_1	f_2	f	f_{avg}	Duration of execution (s)	f_1	f_2	f	f_{avg}	Duration of execution (s)
raxml_am	45.5523	4	7.1108	13.2552	1	5.0931	3	3.2692	6.5115	1.08	0.6368	1	0.7781	0.8486	1
CMS-1500_am	60.8255	8	12.1523	15.2860	5	56.7662	3	7.5903	9.5729	5.36	52.5599	2	3.8533	5.4826	5
nug28-cbb_am	99.7775	4	7.5860	13.9686	10	91.9959	3	6.3321	9.7177	10.7	89.1166	2	3.9122	5.4203	10
nug30-cbb_am	167.3066	7	13.4763	22.8306	20	166.3851	4	10.5603	14.6710	21.5	157.5061	3	5.8878	7.2271	20
raxml_go	28.3623	6	9.0497	12.3524	1	28.2121	3	7.3921	9.0969	1.61	0.5351	1	0.6972	0.7684	1
CMS-1500_go	65.3562	5	9.3652	15.4298	5	53.7482	3	6.6703	9.0542	5.368	46.2074	2	3.8340	5.2608	5
nug28-cbb_go	77.7018	7	12.3838	16.9321	10	79.4214	3	8.1654	10.2845	10.745	77.2029	3	5.7755	5.7760	10
nug30-cbb_go	141.8158	7	12.5855	22.7710	20	154.4203	4	10.2320	13.7650	22.004	137.6620	3	5.8720	7.2010	20
raxml_am	44.8237	6	11.0931	11.0931	180	1.0222	1	1.0039	1.8485	193.404	0.6368	1	0.7781	0.7781	180
CMS-1500_am	73.7056	6	11.4818	12.2660	180	54.1620	2	6.2233	8.1126	193.479	54.6756	1	1.9640	1.9640	180
nug28-cbb_am	92.8217	8	14.8691	16.6128	180	95.6304	2	9.5825	9.8297	193.634	89.4684	2	3.9125	3.9131	180
nug30-cbb_am	178.2715	12	20.6893	22.2003	180	174.4713	3	11.9611	13.6688	194.847	156.4287	3	5.8870	5.8875	180
raxml_go	45.6386	5	9.6928	9.6928	180	10.0532	2	3.1836	3.3291	192.218	0.5351	1	0.6972	0.6972	180
CMS-1500_go	62.9901	8	14.1969	15.1925	180	46.4005	2	7.0035	7.5797	193.569	46.0908	2	3.8336	3.8337	180
nug28-cbb_go	102.7015	9	16.9870	17.1702	180	77.5516	3	9.0945	9.0977	195.795	77.2404	2	3.8990	3.8990	180
nug30-cbb_go	150.7867	8	15.3137	17.4729	180	151.8142	2	9.6737	12.5265	20.0483	137.7398	3	5.8721	5.8721	180

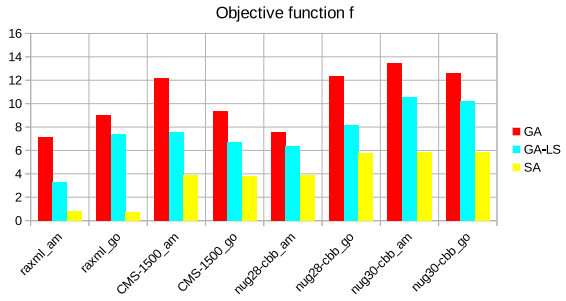

FIG. 4.1. *Comparison of the objective function values found by the GA, the GA-LS hybridisation and the SA.*

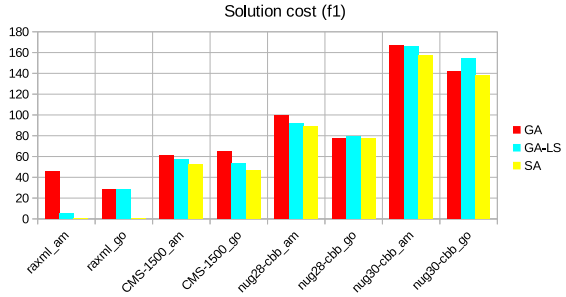

FIG. 4.2. *Comparison of payment costs of solutions found by the GA, the GA-LS hybridisation and the SA.*

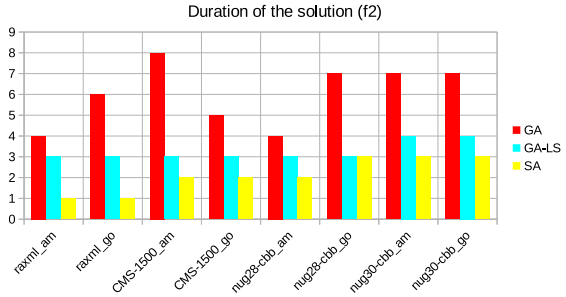

FIG. 4.3. *Comparison of the durations of the solutions found by the GA, the GA-LS hybridisation and the SA.*

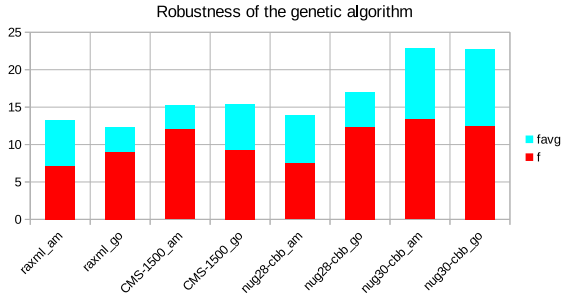

FIG. 4.4. *Robustness of the GA verified by the comparison between the best solution found (f) and the average of the solutions of the ten tests carried out (f_{moy}).*

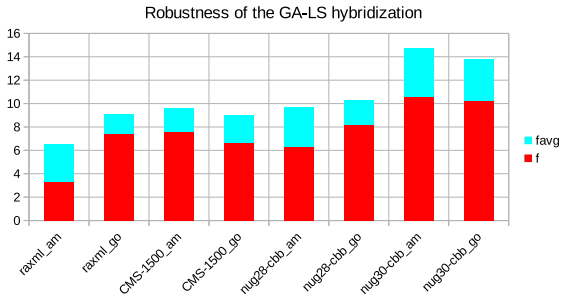

FIG. 4.5. *Robustness of the GA-LS hybridisation verified by the comparison between the best solution found (f) and the average of the solutions of the ten tests carried out (f_{avg}).*

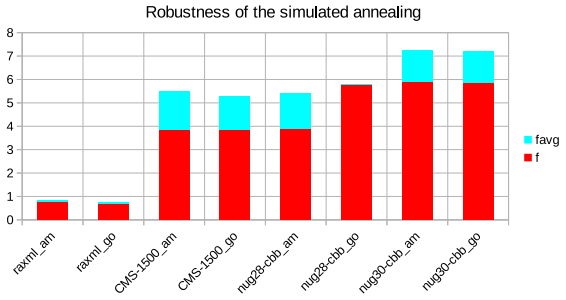

FIG. 4.6. *Robustness of the SA verified by the comparison between the best solution found (f) and the average of the solutions of the ten tests carried out (f_{avg}).*

metaheuristics, namely the Genetic Algorithm, the hybridisation between the Genetic Algorithm and Local Search, and the Simulated Annealing. To compare between these approaches, we launched simulations on different sizes applications. The obtained results showed that the Simulated Annealing finds more optimal solutions than those proposed by the Genetic Algorithm and the hybridisation between the Genetic Algorithm and Local Search.

In the future, we plan to extend our work by comparing the Simulated Annealing with other metaheuristics, and test them in a different context such as the multi-cloud.

REFERENCES

[1] *https://aws.amazon.com/fr/ec2/*, Oct. 2017.
[2] *https://cloud.google.com/compute/*, Oct. 2017.
[3] *https://azure.microsoft.com/fr-fr/*, Feb. 2020.
[4] *https://cloud.google.com/appengine/*, Feb. 2020.
[5] *https://gsuite.google.fr/intl/fr/*, Feb. 2020.
[6] *https://www.heroku.com/*, Feb. 2020.
[7] *https://www.office.com/*, Feb. 2020.
[8] C. BLUM AND A. ROLI, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM computing surveys (CSUR), 35 (2003), pp. 268–308.
[9] C. COLLABORATION, S. CHATRCHYAN, G. HMAYAKYAN, V. KHACHATRYAN, A. SIRUNYAN, W. ADAM, T. BAUER, T. BERGAUER, H. BERGAUER, M. DRAGICEVIC, ET AL., *The cms experiment at the cern lhc*, 2008.
[10] R. D. C. COUTINHO, L. M. DRUMMOND, AND Y. FROTA, *Optimization of a cloud resource management problem from a consumer perspective*, in European Conference on Parallel Processing, Springer, 2013, pp. 218–227.
[11] J. DRÉO, A. PÉTROWSKI, P. SIARRY, AND E. TAILLARD, *Métaheuristiques pour l'optimisation difficile*, 2003.
[12] M. GABLI, E. M. JAARA, AND E. B. MERMRI, *A genetic algorithm approach for an equitable treatment of objective functions in multi-objective optimization problems.*, IAENG International Journal of Computer Science, 41 (2014).
[13] F. GLOVER AND M. LAGUNA, *Tabu search kluwer academic*, Boston, Texas, (1997).

[14] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning, addison-wesley, reading, ma, 1989*, NN Schraudolph and J, 3 (1989).
[15] A. D. Goncalves, L. M. Drummond, A. A. Pessoa, and P. Hahn, *Improving lower bounds for the quadratic assignment problem by applying a distributed dual ascent algorithm*, arXiv preprint arXiv:1304.0267, (2013).
[16] J. H. Holland, *Outline for a logical theory of adaptive systems*, Journal of the ACM (JACM), 9 (1962), pp. 297–314.
[17] H. H. Hoos and T. Stützle, *Stochastic local search: Foundations and applications*, Elsevier, 2004.
[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, science, 220 (1983), pp. 671–680.
[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, The journal of chemical physics, 21 (1953), pp. 1087–1092.
[20] A. Mokhtari, M. Azizi, and M. Gabli, *Optimizing management of cloud resources towards best performance for applications execution*, in 2017 First International Conference on Embedded & Distributed Systems (EDiS), IEEE, 2017, pp. 1–5.
[21] ———, *Multi-cloud resources optimization for users applications execution*, in International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning, Springer, 2018, pp. 588–593.
[22] ———, *A fuzzy dynamic approach to manage optimally the cloud resources*, in International Conference on Innovative Research in Applied Science, Engineering and Technology, IEEE, 2020.
[23] J. Silberholz and B. Golden, *Comparison of metaheuristics*, in Handbook of metaheuristics, Springer, 2010, pp. 625–640.
[24] A. Stamatakis, *Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models*, Bioinformatics, 22 (2006), pp. 2688–2690.
[25] S.H.H. Madni, M.S.A. Latiff, Y. Coulibaly and S.I.M. Abdulhamid, *An appraisal of meta-heuristic resource allocation techniques for IaaS cloud*, Indian Journal of Science and Technology, 2016, 9(4), 1-14.
[26] U.K. Jena, P.K. Das and M.R.Kabat, *Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment*, Journal of King Saud University-Computer and Information Sciences, 2020.

Edited by: Dana Petcu
Received: Aug 14, 2020
Accepted: Dec 6, 2020