Reputation-guided Evolutionary Scheduling Algorithm for Independent Tasks in inter-Clouds Environments

Florin Pop[1], Ciprian Dobre[1*], Valentin Cristea[1], Nik Bessis[2], Fatos Xhafa[3], Leonard Barolli[4]

[1] University Politehnica of Bucharest, Romania
Emails: florin.pop@cs.pub.ro, ciprian.dobre@cs.pub.ro, valentin.cristea@cs.pub.ro (*Corresponding Author)
[2] University of Derby, United Kingdom
Email: N.Bessis@derby.ac.uk
[3] Universitat Politecnica de Catalunya, Barcelona, Spain
Email: fatos@lsi.upc.edu
[4] Fukuoka Institute of Technology (FIT), Japan
Email: barolli@fit.ac.jp

Abstract: Self-Adaptation provides software with flexibility in terms of the different behaviours (configurations) it incorporates and the autonomous or semi-autonomous ability to switch between these behaviours to maintain and maximize its quality in response to changes. For Clouds it becomes important to accommodate uncertainty about clients and the evolving nature of their business and IT worlds: their profiles and skills, competitive technology and business, the devices and network accesses they use, etc. To empower Clouds with ability to capture and respond to the quality feedback, provided by users at runtime, we propose a reputation guided genetic scheduling algorithm for independent tasks. Current resource management services consider evolutionary strategies in order to improve the performance on resource allocation procedures or tasks scheduling algorithms - but they fail to consider the user as part of the scheduling process. Evolutionary computing offers different methods to solve NP-hard problems, finding a near-optimal solution. In this paper we extended our previous work with new optimization heuristics for the problem of scheduling. We show how reputation is considered as an optimization metric analyze how our considered metrics can be considered as upper bounds for others in the optimization algorithm. By experimental comparison, we show that our optimization techniques can be hybridized for optimized results.

Keywords: Scheduling Algorithms; Cloud Computing; Optimization Metrics; Reputation; Evolutionary Computing

Reference to this paper should be made as follows: Florin Pop, Ciprian Dobre, Valentin Cristea, Nik Bessis, Fatos Xhafa, Leonard Barolli. (xxxx) 'The corporate environmental disclosures on the internet: the case of IBEX 35 Spanish companies', *International Journal of Web and Grid Services (IJWGS)*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: *Florin Pop*, PhD, is Associate Professor within the Computer Science Department and also an active member of Distributed System Laboratory in University Politehnica of Bucharest. His interests are on resource management and task scheduling. His expertise is on optimisation based on bio-inspired

techniques received three Best Papers Awards, IBM Assistantship Award and IBM Faculty Award. He is member of the IEEE and ACM professional organizations.

Ciprian Dobre received a PhD in Computer Science at University Politehnica of Bucharest. His main research interests are oriented on Modeling and Simulation, Grid Computing, Monitoring Large Scale Distributed Systems, Parallel and Distributed Algorithms. His research activities were awarded with the Innovations in Networking Award in 2008 by CENIC and three Best Paper Awards. He is member of the IEEE and ACM professional organizations.

Valentin Cristea is Professor of the Computer Science Department of University Politehnica of Bucharest. His main fields of expertise are Large Scale Distributed Systems and e-Services. He teaches courses, supervises PhD students, leads projects, and is active in research of these topics. He is member of the IEEE and ACM professional organizations.

Nik Bessis is currently Head of Distributed and Intelligent Systems (DISYS) research group, a full Professor and a Chair of Computer Science in the School of Computing and Mathematics at University of Derby, UK. His research interest is the analysis, research, and delivery of user-led developments with regard to data integration, annotation, and data push methods and services in distributed dynamic environments.

Fatos Xhafa is Hab. Full Professor and holds a permanent position of Professor Titular at the Department of LSI and member of the ALBCOM Research Group. His current research interest include parallel and distributed algorithms, combinatorial optimization, approximation and meta-heuristics, networking systems, distributed programming, Grid and P2P computing.

Leonard Barolli is full Professor in the Department of Information and Communication Engineering, Fukuoka Institute of Technology, Japan. He has published more than 200 papers in refereed journals and international conference proceedings. He has served as a Guest Editor for many journals. His research interests include high-speed networks, grid computing, P2P, and ad-hoc and sensor networks. He is a member of IEEE, IPSJ and SOFT.

1 Introduction

The current approaches to software adaptation have critical limitations when applied on software in a cloud environment. The source of the reasons is that Cloud engineers cannot be certain of the nature of client organizations and their users' who will be eventually using provided software or service, and their evolution over time. The openness feature of the cloud makes it hard, and almost impossible, for software engineers to make assumptions and decisions on how service-adaptation should be planned.

Such approaches for software adaptation have critical limitations when applied on software in a Cloud environment. The source of the reasons is that Cloud engineers cannot be certain of the nature of client organizations and their users' who will be eventually using provided software or service, and their evolution over time. The openness feature of the Cloud makes it hard, and almost impossible, for software engineers to make assumptions and decisions on how service-adaptation should be planned. Thus, for Clouds it becomes important to accommodate uncertainty about clients and the evolving nature of their business and IT worlds: their profiles and skills, competitive technology and business, the devices and network accesses they use, etc. Here we propose a solution to empower Clouds with

the ability to capture and respond to the quality feedback, provided by users at runtime. We propose the use of the *reputation* quality metric associated with a Cloud service [1].

Service-adaptation based on reputation can lead to the increase of the trust associated with Cloud services. Since various Cloud providers can provide the basic services at different levels and with various pricing models, dishonest providers could claim arbitrary QoS properties to attract interested parties. The standard way to prevent this is to allow users to evaluate a service and provide feedback.

However, the feedback mechanism has to ensure that false ratings, for example, badmouthing about a competitor's service or pushing one's own rating level by fake reports or collusion with other malicious parties, can be detected and dealt with. Consequently, a good service discovery engine would have to take into account not only the functional suitability of services but also their prospective quality offered to end-users by assessing the trustworthiness of both providers and consumer reports. Solutions for QoS-based service selection and ranking algorithms for distributed service discovery approaches already exist [1]. Here, we extend on this and propose the use of reputation for Cloud adaptation: services and middleware level executed tasks are intelligently selected and aggregated based on their reputation, considering an environment where multiple Cloud providers can put together their resources. Thus, we propose a reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments.

Braun et al. [4] previously showed that the most effective non-evolutionary method for scheduling independent tasks is the min-min heuristic. This method builds a schedule by successively searching the list of tasks not yet assigned to find the task with the minimum completion time then assigns the task to the schedule and then repeat the process until all tasks are assigned. Another heuristic is min-max that builds a schedule by assigning tasks in the order they are found in the task queue. Each task in turn is paired with the machine for which the incremental cost is minimum unless that assignment would increase the makespan, in which case the task is assigned to the machine for which incremental maxspan is minimum [5]. The randomized min-span algorithm [6] uses a succession of randomized ordering of the task list with the min-span heuristic to search for an optimal solution. The opportunistic load-balancing (OLB) heuristic picks one task arbitrary from the group of tasks and assigns it to the next machine that is expected to be available and do not consider the execution time of the task on that machine, leading to a big maxspan. The big advantage of this method is the simplicity and that it keeps the machines busy improving processor utilization rate. The minimum execution time (MET) heuristic assigns each task randomly chosen to the machine that has the least expected execution time for that task. This may lead to imbalance load between machines, but gives each task to its best machine. The minimum completion time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected time of completion for that task [7].

Such optimizations of the scheduling process are designed to provide better selection and allocation of resources. The optimization is important because the scheduling is a main building block for making Grids and Clouds more available to user communities. The optimization metrics for task scheduling and resource management are the main subject of this paper. Similar research conducted by the authors of this papers is presented in [8] and [9]. Different solutions for multi-objective decentralized control models for tasks assignment in Cloud systems. The transaction in real-time complex system is modeled by tasks which will be scheduled and executed in a distributed system, so a set of specifications and requirements are known [8]. We extend on this, and propose a scheduling solutions where 1) reputation

is considered as an active quality metric in the scheduling decision, and 2) tasks can be executed in an environment composed of multiple Cloud platforms.

The proposed recommendation-based scheduling approach becomes quite important in the Big Data Era. The explosion and profusion of available data in a wide range of application domains rise up new challenges and opportunities in a plethora of disciplines - ranging from science and engineering to biology and business. One major challenge is how to take advantage of the unprecedented scale of data - typically of heterogeneous nature - in order to acquire further insights and knowledge for improving the quality of the offered services. To exploit this new resource, we previously show [10] that we need to scale up and scale out both our infrastructures and standard techniques. Our society is already data-rich, but the question remains whether or not we have the conceptual tools to handle it. Big Data is the next frontier for innovation, competition, and productivity, and many solutions continue to appear, partly supported by the considerable enthusiasm around the MapReduce (MR) paradigm for large-scale data analysis. Our scheduling approach is well tailored around the independent task execution assumed by the MR paradigm. In addition, the scheduling algorithm could improve the performance of the MR execution platforms by incorporating the user's opinion (i.e., the reputation) as an active metric for adaptation of the Cloud execution environment - with great performance advantages, as our experimental results show, especially in heterogeneous inter-Clouds.

In this paper we extended our previous work in [11] with new optimization heuristics for the problem of scheduling according with [3]. In particular, we show how reputation is considered as an optimization metric. In particular, we show that our considered metrics can be considered upper bounds for others in the optimization algorithm, and present a general model for an evolutionary algorithm (based on genetic algorithms) and the appropriate metrics for solution selection. By experimental comparison, we show that our optimization techniques can be hybridized for optimized results.

The rest of the paper is structured as following: Section 2 presents the proposed solution based on Evolutionary Genetic Algorithm, and Section 3 presents the optimization criteria for the evolutionary scheduling algorithms. Section 4 describes the reputation influence, Section 5 presents the applicability in inter-Clouds environments, and Section 6 analyzes the experimental results. Finally, in Section 7 we present the conclusions.

2 Proposed Solution based on Evolutionary Computing

A genetic algorithm (GA) is a search algorithm based on the principle of evolution and natural genetics. GA provides solutions for the scheduling problem with both independent and interdependent tasks. The first step of the GA is choosing the right representation of each gene. For independent tasks the most common representation is a pear (T_i, R_j) showing that task T_i will be executed on processor R_j. The execution order of tasks on the same processor is given by the position of the gene in the chromosome. Another representation uses a vector in which each character is a mapping between a task and processor. Each character represents the unique identification number of a task and the -1 character is used to delimit the processor queues for each different processor. This representation has a length of $N + M - 1$, where N is the number of tasks in the batch and M is the total number of processors, according with [12]. Zomaya proposed in [13] a two dimensional representation. The first dimension is equal to the number of processors and the second dimension holds the tasks which are to run on that processor. The disadvantage of the representation is that

the crossover operator cannot work on two-dimensional presentations, so the string must be compacted in a one-dimension array by connecting the two-dimensional array end-to-end to form a long string.

In our approach we will propose the gene encoding as follow:

$$[T_i, R_j, Rep(R_j)] \tag{1}$$

where $Rep(R_j)$ represents the reputation evaluation for resource R_j. This value can be changed in the mutation phase according with dynamic changes in the environments. We will discuss in details the reputation influence in Section 4.

For the set of tasks $T = \{T_1, \ldots, T_N\}$ and the set of resources $R = \{R_1, \ldots, R_M\}$ we have a chromosome representation:

$$Ch(T, R) = \{[T_i, R_j, Rep(T_j)] | T_i \in T, R_j \in R\} \tag{2}$$

and $|Ch(T, R)| = |T| = N$, meaning that all tasks are scheduled. For $N < M$ we will have several resource unused, for $N > M$ several resource will execute more than one task and for the special case when $N = M$ it will be possible to have one-to-one mapping (perfect load-balancing). In general, we have $N > M$.

Crossover represents reproduction in the GA. After the selection stage, some individuals are chosen to cross part of them. It's an important operation that creates based offspring in each generation. Crossover is not usually applied to all individuals. For that is defined a crossover probability that is typically between 0.5 and 1.0. There are three classic crossover types. The simplest form is the *single point crossover* which takes two individuals and cuts their chromosome strings at some randomly point to produce two 'front' and two 'end' segments. The end segments are swapped over to produce two new full length chromosomes. In this way the two offspring inherit some genes from each parent. There are crossover algorithms that involve more than one cut point, called *multiple crossover points*. Multiple crossover takes n random crossover points and exchange components of a chromosome between successive cut points to produce two new offspring. Having more crossover points has an advantage in that the space may be searched more thoroughly. *Uniform crossover* is different from the above two methods. Each gene in the new offspring is created by copying the corresponding gene from one of the parents according to a mask generated randomly. The crossover algorithms can be particularized for the scheduling problem.

In our proposed solution, considering the chromosome encoding $[T_i, R_j, Rep(R_j)]$, the crossover described above can be applied on the strings that compose the encoding by swapping over only the second part of the string, the processor on which a task is scheduled.

Mutation ensures that the probability of finding the optimal solution is never zero. It also ensures the recovery of the good material genetic that may be lost through selection and crossover. Mutation is applied to the new chromosomes with a certain probability. The mutation can be static and is applied to the entire generation with a probability in the range $[0.1, 1.0]$, and dynamic. In the dynamic case, the mutation probability increases linearly when the population converges, in order to find new variations. Or the mutation probability can decrease exponentially during a run as the population converges. Increasing and decreasing the probability avoid local optima. In a fitness adaptive algorithm, the mutation does not have the same probability for all the individuals in a generation. Low-fitness chromosomes have an increased probability, in order to explore the search space. High-fitness chromosomes have a decreased probability of mutation in order to focus on a

specific region of possible optimum, without losing fit individuals. Variations of mutation operator change the position of genes in the chromosome: *Order-based Mutation* â randomly selects two genes of a chromosome and interchanges their positions, *Sublist-based Mutation* â randomly selects two points in a chromosome and reverses the order of genes between these points. For task scheduling the above methods don't move the task from one processor to another. For this, *partial-gene mutation* may be used, that randomly selects a chromosome and changes a randomly gene by assigning the task to a new processor where it has earliest start time [13] proposes a *Swap-Gene Mutation*. This method selects two processors, and then randomly selects a task on each processor. The tasks are interchanged between processors if they have similar property (example, the same priority). This method can be changed to consider the topological level on the tasks, according with [14]. Here, it is selected a processor that has assigned at least two jobs with the same topological level and selects, also randomly, two tasks with the same topological level assigned to the processor and the tasks are interchanged if the first task to execute as fewer children that the second one. The *Additive Mutation*, introduced by [13], randomly selects two processors, and then randomly selects a task on the first processor. A starting point is randomly selected, after which the processor sub-string is searched for insertion. The task must be inserted such that the priory rules are still fulfilled. Mutation enlarges the search space to the vicinity of the population by randomly altering genes. This way the tendency is to evolve through a global rather than a local optimum. Although is considered that the crossover is the most important operator in GA, on long runs the mutation generally finds better solutions than crossover alone does.

The evaluation of a chromosome is the most critical portion of a genetic algorithm. The main objective is to achieve minimum execution time, maximum processors utilization and a well-balanced load across all processors. The most known metric in fitness computing is the maxspan of the task schedule that is the largest task completion time among all the processors in the system. For each processor, the total completion time represents the sum of all completion times for the tasks assigned to R_j in the current schedule, or have already assigned in the previous schedule, but not yet executed.

The selection is done using the roulette wheel principle. Thus, the better the fitness of an individual is, then the higher odds to be selected. The slots of the roulette wheel must be determined based on the probabilities of individual strings surviving into the next generation. These probabilities are calculated by dividing the fitness of each chromosome by the sum of all fitness functions from the population.

The operators used in this proposed solution are: (i) One-Point Crossover and Order-based Mutation and (ii) Two-Point Crossover and Partial-Gene Mutation.

The fitness function is can be defined according with different optimization factors (f_i) and different weights (w_i). There are multiple variants of building the fitness function. The general is to define fitness as:

$$F(Ch(T,R)) = \sum_i w_i f_i(Ch(T,R)) \tag{3}$$

where $\sum w_i = 1$, or, in a specific way, using the same weights, but as a product form:

$$F(Ch(T,R)) = \prod_i f_i(Ch(T,R)) \tag{4}$$

Considering load-balancing as a way to measure the optimization impact for providers and minimum execution time as a metric for user performance, we can define a simple fitness function as:

$$F(Ch(T,R)) = \frac{minspan}{maxspan} \times \frac{1}{M} \sum_{j=1}^{M} \frac{completion\_time(R_j)}{maxspan} \times f(Rep(R))) \quad (5)$$

where $minspan$ is the minimum execution time of the processors and f is a function that depends on reputation estimation. We used the following form for f function:

$$f(Rep(R)) = \max_{R_j \in Ch(T,R)} \left\{ \frac{Rep(R_j)}{\max_{1 \leq j \leq M} \{Rep(R_j)\}} \right\} \quad (6)$$

As we can see, it is possible that the $f(Rep(R))$ to be equal to 1 if the best resources is selected in the current schedule. We will prove, based on experiments, that this function will do not affect the load-balancing for resources in a Cloud.

The Evolutionary scheme in a general form is described by Algorithm 1.

Algorithm 1 Evolutionary Algorithm Scheme

Step 1: Setting the parameters:
1. Load the task descriptions;
2. Initiate the population size: `pop_size`;
3. `crossover probability` $\leftarrow p_c$;
4. `mutation probability` $\leftarrow p_m$;
5. `maximum generation` $\leftarrow maxgen$;
6. $gen = 0$.

Step 2: Initialization
Generate pop_size chromosomes using precedence resolutions and/or by random, depending on task descriptions.
while $gen \neq maxgen$ **do**
Step 3: Evaluate
Calculate the fitness value of each chromosome based on specific criteria.
Step 4: Selection
Select pop_size chromosomes from the parents and offspring for the next generation.
Step 5: Crossover/Clone
Perform the crossover (for genetic algorithms) or clone operation (for immune algorithms) on the chromosomes selected with probability p_c.
Step 6: Mutation
Perform the swap/move mutation on chromosomes selected with probability p_m.
Step 7: Elitism or Negative Selection
Preserve the best chromosome from the previous generation (elitism) or eliminate the bests and the worsts chromosome (negative selection).
Step 8: $gen = gen + 1$;
end while
Step 9: Preserve the best chromosome as final solution.

3 Optimization Criteria and Optimization Metrics

One of the main aims of optimization metrics in distributed systems is to reduce the energy consumption. To find the minimum energy consumption [15] consider to combine an architecture that uses Internet layers with a physical layer operating at the fundamental limit or energy dissipation. They consider that for large networks, this layered approach is more feasible than an analysis based on information theory. On the users side mobile devices (laptops, smart phones, tablets, embedded boards, robots) can serve as clients for Cloud Computing and are based for Mobile Cloud. Here, important issues include optimizing the scheduling and transport schemes, access management, and application optimization, for mobile devices to achieve energy saving, according with [16].

In [17] a series of green computing systems are presented, like: evolutionary Green Computing solutions for distributed Cyber-Physical Systems, energy-aware provisioning of HPC services with virtualised web services, energy and security awareness in evolutionary driven Grid Scheduling, power consumption constrained task scheduling using enhanced genetic algorithms, exploiting multi-objective evolutionary algorithms for designing energy-efficient solutions to data compression and node localization in wireless sensor networks. There approached prove that, in general, the green computing systems considers large and massive systems like Cloud, where saving energy will contribute to ecosystems, but also considers pervasive (mobile) systems where saving energy means longer life for devices and systems. Task scheduling and resource management represent and important aspects for Green Computing systems building.

For optimization criteria we consider the set of tasks $\{T_i\}_{1 \leq i \leq n}$, each task having at least two characteristics $T_i = (C_i, d_i)$, where C_i is the finishing (completion) time for T_i and d_i is the specified deadline for T_i. In this approach C_i is a dynamic characteristic and is not known apriori and usually C_i depends on resource that will execute T_i. For the environment we consider a set of resources having the capabilities R_j^i - processing power consumed by processor j for T_i. We also consider the following optimization metrics (objectives), defined in many form in literature, like in [3]:

- **makespan** (maximum completion time): $C_{max} = \max_i \{C_i\}$.
- **flow time** (total execution time): $FlowTime = \sum_i C_i$.
- **lateness**: $L_i = C_i - d_i$.
- **earliness**: $E_i = \max\{0, d_i - C_i\}$.
- **tardiness**: $T_i = \max\{0, C_i - d_i\}$.
- **absolute deviation**: $D_i = |C_i - d_i|$.
- **squared deviation**: $S_i = (C_i - d_i)^2 = D_i^2$.
- **unit penalty**: $U_i = \begin{cases} 0 & \text{if } C_i < d_i \ (T_i = 0) \\ 1 & \text{otherwise} \end{cases}$

The weighted criteria are introduce based on the the above definitions. Having the weight vector $w = (w_1, w_1, \cdots, w_n) \in R^n$, usually with $\sum_i w_i = 1$ the following criteria are defined:

- **weighted FlowTime**: $FlowTime(w) = \sum_i w_i C_i$

- **weighted objective**: if $X_i \in \{T_i, U_i, D_i, S_i, E_i\}$, we define $X_{(w)} = \sum_i w_i X_i$. The sum objectives is a particular case with $w_i = \frac{1}{n}$, so $X_{(1/n)} = \frac{1}{n}\sum_i X_i$. In this case, the $\frac{1}{n}$ is not an important factor in optimization, so it is possible to be ignored and the weighted objective transforms in **sum objective**.

- **maximum lateness**: $L_{max} = \max_i \{L_i\}$.

All criteria depending on d_i are support for optimization in real-time systems, where deadlines are very important. Several specific scheduling problems are: $1|prec; p_i = 1|\sum_i w_i C_i$ - optimization of uniform tasks with dependencies on a single machine; $P_2||C_{max}$ - optimization problem on two identical processors; $R_2|pmtn; intree|C_{max}$ - optimization of task scheduling on two dedicated uniform machines with tree dependencies and preemption.

The following results are very important for optimization criteria and metrics:

- $C_{max} \leq FlowTime \leq nC_{max}$. This results prove that the optimization under the C_{max} or $FlowTime$ are equivalent. The solution presented in [18] considers a genetic algorithm with the fitness function defined as: $fitness = \lambda C_{max} + (1 - \lambda)FlowTime$, where λ has been a priori fixed after a preliminary tuning process. This approach is oriented only on non-real-time systems and only on statical scheduling.

- $L_{max} \leq threshold$ is equivalent with $\sum_i T_i \leq 0$ and $\sum_i U_i \leq 0$. The results is presents in [3] and prove that the optimization in real-time systems, depending on deadlines d_i can be performed using any criteria.

Evolutionary algorithms operates through a simple cycle of stages: creation of population of chromosome, evaluation of each chromosome (according with a specific fitness function), selection of the best chromosome for genetic algorithms or eliminate the best and the worst chromosomes and reproduction (crossover or clone) to create a new population.

The number of genes and their values in each chromosome depend on the problem specification. We considered that the number of genes of each chromosome is equal to the number of the nodes (tasks) in the scheduling problem and the gene values demonstrate the scheduling priority of the related task to the node (each chromosome shows a possible schedule), where the higher priority means that task must be executed early. In the basic genetic algorithm the initial population is generated randomly, which can cause to generate more bad results. To avoid the generation of non-optimal results, heuristic approach along with precedence resolution can be applied to generate the initial population that gives better results in terms of quality of solutions (see [19]).

For the evolutionary algorithms we extend the tasks characteristics with the processing time p_i and arrival time a_i so we have $T_i = (C_i, d_i, p_i, a_i)$. In this model we have the following constrain $p_i \leq C_i - a_i$. Let's ch to be a chromosome in the population. We consider the following form of fitness function:

- **Best finishing time**:

$$f_1(ch) = \max_{ch}(C_{max}(ch)) - C_{max}(ch) + \epsilon = C_{max} - C_{max}(ch) + \epsilon \qquad (7)$$

where $C_{max}(ch) = \max_{i \in ch} \{C_i\}$. ϵ is a value used to avoid that the fitness function is zero, so it can be choose as $\epsilon = fC_{max}$ (a fraction from C_{max}), so we have:

$$f_1(ch) = (1 + f)C_{max} - C_{max}(ch). \qquad (8)$$

- **Finishing time**:

$$f_2(ch) = \max_{ch}(\tau_{max}(ch)) - \tau_{max}(ch) + \epsilon, \tag{9}$$

where $\tau_{max}(ch) = \max_{j,i \in ch}\{\tau_{ij}\}$ and $\tau_{ij} = p_i/R_j^i$. Considering the hypothesis $\max_j\{\tau_{ij}\} \leq C_i$, we have the following result: $f_2(ch) \leq f_1(ch)$.

- **Personal makespan**, the largest task completion time among all processors for a specific schedule:

$$f_3(ch) = C_{max}(ch). \tag{10}$$

- **Acceptable processor queues**. Each processor queue is checked to see if assigning all the tasks on the processor queues will overload or under load the processors. A threshold determines this. If the task completion time of a processor is within the thresholds, this processor queue is acceptable.

$$f_4 = \frac{|Queues_{acceptable}|}{|proc|}. \tag{11}$$

- **Average tardiness of jobs in the schedule**:

$$f_5(ch) = \frac{1}{n}\sum_{i \in ch} w_i T_i. \tag{12}$$

- **Number of tardy jobs**:

$$f_6(ch) = \sum_{i \in ch} T_i. \tag{13}$$

- **Total throughput time**:

$$f_7(ch) = \sum_{i \in ch}(C_i - a_i) \quad or \quad f_7(ch) = \sum_{i \in ch} p_i. \tag{14}$$

- **Load balancing**: is a measure of the uniformity of the tasks disposal on the processors, with the purpose to obtain similar execution time on all processors and reduce overloading. Load balancing is computed with the formula:

$$f_8(ch) = 1 - \frac{1}{wp}\sqrt{\sum_{j \in ch}(wp - wp_j)^2}, \tag{15}$$

where wp_j is the processor j utilization rate, and j is used in schedule represented by chromosome ch and wp is the average value of utilization rate for all processors.

4 Reputation Influence

We introduced a measure of trust that previous consumers accessing a resource have about execution of the service in the context of a resource provider. This is a Reputation ranking given by the client. In order for this ranking not to be falsely reported data the system will reject any data that does not follow the normal distribution of the reputation information collected by the system. The ranking information rejection will start after collecting a given number of reputation samples and the normal distribution will be re-adjusted in time by the system. In addition to these metrics, the system is aware of the business context of web services, and therefore assumes that the amount of money that a service requester has to pay to the service provider for accessing a specific hosted service (Execution Price) and the percentage of the original execution price that will be refunded when the service provider cannot honor the committed service or deliver the ordered commodity (Compensation Rate) will be supplied by the service in question. One important aspect, not taken into consideration in the previous work on the resources selection methods, is related with the ability of a service provider to process requests within a given execution time constraint when multiple consumers are accessing its exposed services. When selecting the "best" available resource, the centralized selection algorithm takes into consideration the execution time as a function of the resource's spare capacity. Using this for the resource selection gives better results over time as it gathers information on the Scalability of the resource's execution in the hosting environment and consequently avoids the selection of the same provider for consecutive calls that would result in provider's "flooding" and bad execution performance. This approach encourages the uniform distribution of the requests to all available replicas after the system starts. In time, the measured scalability of the replicas impacts the distribution of the requests [20].

The reputation consider the following set of metrics (r_k) oriented in QoS and business context:

- $r_1(R_j)$: Execution Time on R_j (QoS)
- $r_2(R_j)$: Transmission Time to R_j (QoS)
- $r_3(R_j)$: Spare Capacity of R_j (QoS)
- $r_4(R_j)$: Execution Price on R_j (Business)
- $r_5(R_j)$: Compensation Rate of R_j (Business)

Finally, the cumulative metric (based on $|r| = 5$ metrics) for reputation is computed using as follows:

$$Rep(R_j) = \sum_{k=1}^{|r|} w_k r_k(R_j) \tag{16}$$

where w_k is the weight of the metric k. The metrics weights will be adjusted during the experimental phase. Also, we see the usage of the weight building a personalized Scheduling system in a Cloud based on the users' sensitivity to specific metrics.

5 Inter-Clouds Applicability

Cloud Computing are one of the fastest evolving paradigm in the domain of Computer Science. Whether one wants to provide a simple file transfer service that consumes an insignificant amount of resources and time, or a parallel and distributed algorithm that defines a weather prediction model that requires high computing power or even a very strict and secure banking service, its implementation by means of a service has a great number of advantages. Tasks execution can address one Cloud or multiple Clouds, depending on users' requirements. So, hybrid Clouds will be considered and inter-Clods environments become the fundamental platform for tasks execution.

Right now people are choosing cloud providers based on mostly hearsay, and most of them are simply going with Amazon because it has it all and you can customize it to your heart's content. The problem with this approach is that it doesn't encourage competition; nonetheless in the last few years powerful competitors have arrived, some like Rackspace that has actually cornered 4% of the market (see [21]). This is also because they are offering better pricing than the Amazon here in EU, but so far the idea is that people simply don't have a way to look at all the offers in one place and make an educated decision. This is the reason why later last year, a new start-up has arrived on the marked, Cloudorado, they are actually parsing all the offers of the providers and then using the methodology described above to compare offers with respect to what EC2 is offering, for your needs, according with [22]. This is a good start, but it will serve only as comparative tool for people, because they are supposed to know how long their applications will run and how resource intensive they'll be.

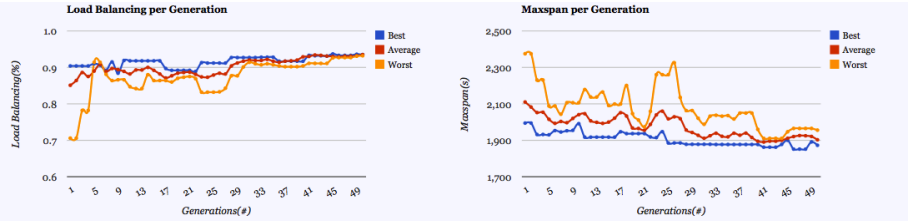
These aspects motivate the use of reputation based method for task scheduling using evolutionary algorithms (like GA).

6 Experimental Results

For the first part of the experiments, the objective is to create a testing environment that can help to identify the genetic algorithm parameters that gives a near-optimal solution for the scheduler using the reputation function computed in a transparent way behind the Scheduling System. We implemented a simple framework for tasks submission and scheduling on the top of different virtual resources.

This framework was previously used to evaluate the evolution of the genetic algorithm and to find the optimal values for the mutation and crossover probabilities and also the mutation and crossover type. The best results were obtained for crossover probability of 0.7 and mutation probability of 0.3. Regarding the algorithms operators, the best choice was the one-point crossover or two-point crossover and partial-gene mutation (see Figure 1 and 2).

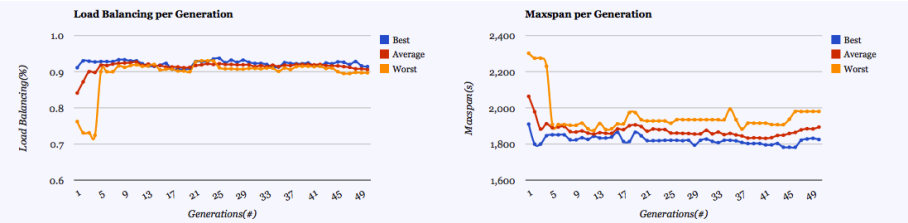
The testing parameters for the scheduler are the load balancing with the objective to obtain an workload well-balanced across all VMs (processors), so a value closer to 1.0 (see Figure 1), and the maxspan with the objective to obtain a low value for the completion time (see Figure 2). These experiments show that, using the resource selection based on reputation function with weight for execution time equal to 1, we can obtain in the same time a good load-balancing (maximizing provider profit) and minimizing the maxspan (minimizing users' costs).



Type	Best off. load	Average load	Worst off. load
Min Value	0.88	0.85	0.71
Max Value	0.94	0.93	0.93
Average Value	0.92	0.9	0.87
Min gen	9	1	1
Max gen	45	50	50

Type	Best off. max	Average maxspan	Worst off. max
Min Value	1851.87	1890.66	1910.84
Max Value	1994.52	2110.11	2374.91
Average Value	1906.24	1974.17	2079.94
Min gen	46	41	41
Max gen	1	1	1

Figure 1 Experimental results: 50 generation for GA, crossover probability = 0.5, One-Point Crossover, mutation probability = 0.5, Order-based Mutation, 500 tasks, 10 virtual resources



Type	Best off. load	Average load	Worst off. load
Min Value	0.91	0.84	0.72
Max Value	0.94	0.93	0.93
Average Value	0.92	0.91	0.9
Min gen	19	1	4
Max gen	25	11	22

Type	Best off. max	Average maxspan	Worst off. max
Min Value	1782.31	1831.41	1874.55
Max Value	1909.85	2063.57	2301.74
Average Value	1822.16	1873.63	1954.72
Min gen	44	41	12
Max gen	1	1	1

Figure 2 Experimental results: 50 generation for GA, crossover probability = 0.7, Two-Point Crossover, mutation probability = 0.3, Partial-Gene Mutation, 500 tasks, 10 virtual resources

In the second par of the experiments, we have considered two version of evolutionary scheduling algorithms based in generic algorithms and immune algorithms. We used a framework already contained an implementation of a basic genetic algorithm with C_{max} optimization metric for task scheduling [23]. We called this first version: HGA (Heuristic based Genetic Algorithm).

The second algorithm is a combination between a genetic algorithm and an immune algorithm (GAIIA). In GAIIA, after the selection phase, we choose randomly several chromosomes from population obtained by immune algorithm and insert them into the population generated by genetic algorithm. At the end, the final solution is provided by generic algorithm. The considered fitness function was based on the discussion from Section 3:

$$F(ch) = \frac{\min_{i \in ch} C_i}{C_{max}} \times \sum_{i \in ch} (C_i - a_i) \times \sum_{i \in ch} T_i. \tag{17}$$

We compared the improved algorithm presented here with the basic one. The metric used as an indicator of performance is the makespan. The tests were made for the cases of

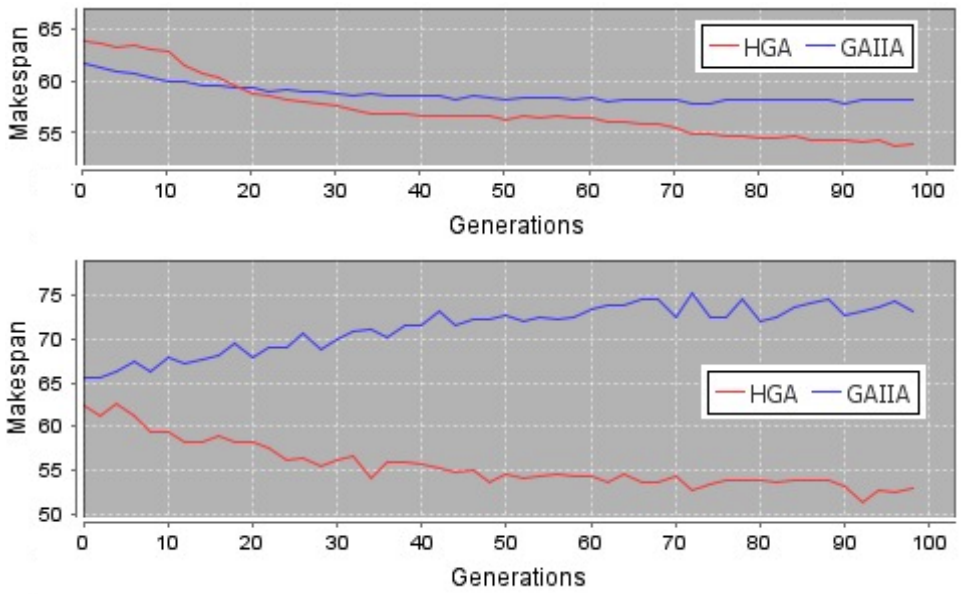


Figure 3 HGA and GAIIA comparison for 40 tasks: 4 and 8 VMs

4 and 8 VMs with 2 task configurations, first one with 40 tasks and the second one with 90 tasks. The results are presented in Figures 3 and 4. The results confirm that after lower number of iteration we can stop the HGA and GAIIA and we obtain a near-optimal solution. On the other hand, a significant difference between HGA and GAIIA can be seen for 8 VMs.

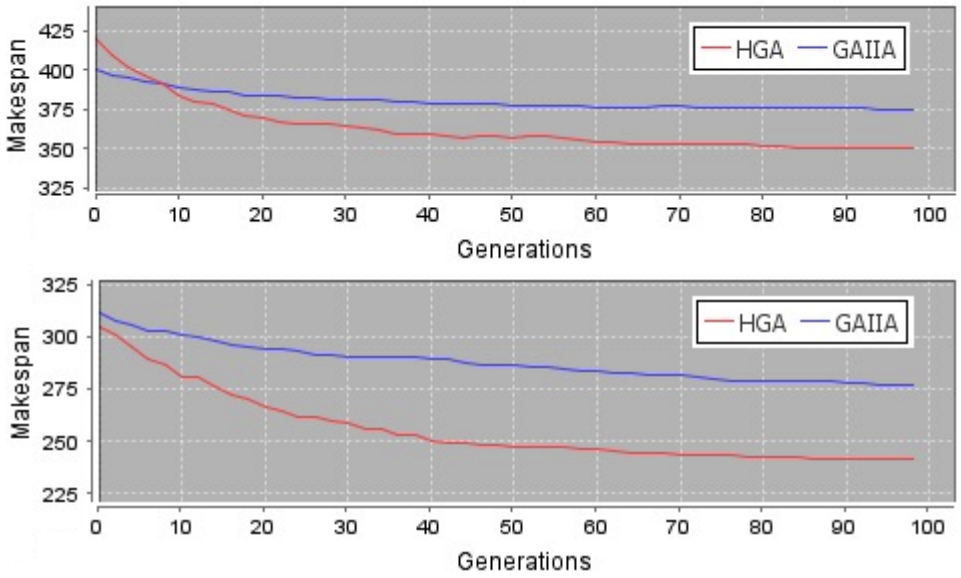


Figure 4 HGA and GAIIA comparison for 90 tasks: 4 and 8 VMs

7 Conclusions

We described in this paper an evolutionary approach for scheduling problem optimization using reputation evaluation in the resources selection phase and several optimization metrics. For the proposed genetic algorithm we moved toward a multi-objective approach that consist of combining the different objective into a weighted sum for reputation function. Also, in this paper we were able to see the evolution and performance factors of the algorithm with different probability for the GA operators (crossover and mutation). W can conclude that, the heuristic algorithms which obtain near-optimal solution in an acceptable interval time are preferred to the back tracking and the dynamic programming. The genetic algorithm is one of the heuristic algorithms which have the high capability to solve the complicated problems like the task scheduling. The heuristics based genetic algorithm proved to be a viable solution for the task scheduling problem on a homogeneous parallel system, performing better than standard genetic algorithms. Future research involves a dynamically change of the weights of the factors that computes the fitness function. In this way, the algorithm can improve his results. The change can be done taking into account information like workload, completion time of the best scheduler from the generation.

Acknowledgements

The research presented in this paper is supported by the following projects: *ERRIC - Empowering Romanian Research on Intelligent Information Technologies*", FP7-REGPOT-2010-1, ID: 264207; "*SideSTEP - Scheduling Methods for Dynamic Distributed Systems: a self-* approach*", (PN-II-CT-RO-FR-2012-1-0084);"; *CyberWater* grant of the Romanian National Authority for Scientific Research, CNDI-UEFISCDI, project number 47/2012.

We would like to thank the reviewers for their time and expertise, constructive comments and valuable insights.

References

[1] Le-Hung Vu, Manfred Hauswirth, and Karl Aberer. Qos-based service selection and ranking with trust and reputation management. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 466–483. Springer, 2005.

[2] Young-Choon Lee and Albert Zomaya. A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems. *Parallel and Distributed Systems, IEEE Transactions on*, 19(9):1215–1223, 2008.

[3] Peter Brucker. *Scheduling algorithms*. Springer, 2007.

[4] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing*, 61(6):810–837, 2001.

[5] Tao Xie and Xiao Qin. Performance evaluation of a new scheduling algorithm for distributed systems with security heterogeneity. *Journal of Parallel and Distributed Computing*, 67(10):1067–1081, 2007.

[6] Wayne F Boyer and Gurdeep S Hura. Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments. *Journal of Parallel and Distributed Computing*, 65(9):1035–1046, 2005.

[7] Joanna Kołodziej and Fatos Xhafa. Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. *Future Generation Computer Systems*, 27(8):1035–1046, 2011.

[8] Florin Pop. Optimization of resource control for transitions in complex systems. *Mathematical Problems in Engineering*, 2012, 2012.

[9] Florin Pop. Heuristics analysis for distributed scheduling using monarc simulation tool. In *ICMS 2012: International Conference on Modeling and Simulation, Zurich, Switzerland*, pages 157–163, 2012.

[10] Ciprian Dobre and Fatos Xhafa. Parallel programming paradigms and frameworks in big data era. *International Journal of Parallel Programming*, pages 1–29, 2013.

[11] Florin Pop, Valentin Cristea, Nik Bessis, and Stelios Sotiriadis. Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 772–776. IEEE, 2013.

[12] Tomasz Janowski and Mathai Joseph. Dynamic scheduling and fault-tolerance: Specification and verification. *Real-Time Systems*, 20(1):51–81, 2001.

[13] Albert Y Zomaya, Chris Ward, and Ben Macey. Genetic scheduling for parallel processor systems: comparative studies and performance issues. *Parallel and Distributed Systems, IEEE Transactions on*, 10(8):795–812, 1999.

[14] Mihai Istin, Florin Pop, and Valentin Cristea. Higa: Hybrid immune-genetic algorithm for dependent task scheduling in large scale distributed systems. In *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on*, pages 282–287. IEEE, 2011.

[15] Kaushik R Chowdhury, Dave Cavalcanti, Mostafa El-Said, Tommaso Mazza, and Chittabrata Ghosh. Modeling and simulation of smart and green computing systems. *Computer*, 45(9):22–23, 2012.

[16] Ivan Stojmenovic. Keynote 1: Mobile cloud and green computing. *Procedia Computer Science*, 10:18–19, 2012.

[17] Samee Ullah Khan, Joanna Kolodziej, Juan Li, and Albert Y Zomaya. *Evolutionary based solutions for green computing*. Springer Publishing Company, Incorporated, 2012.

[18] Javier Carretero, Fatos Xhafa, and Ajith Abraham. Genetic algorithm based schedulers for grid computing systems. *International Journal of Innovative Computing, Information and Control*, 3(6):1–19, 2007.

[19] Kamaljit Kaur, Amit Chhabra, and Gurvinder Singh. Heuristics based genetic algorithm for scheduling static tasks in homogeneous parallel system. *international journal of computer science and security*, 4(2):183–198, 2010.

[20] O Achim, Florin Pop, and Valentin Cristea. Reputation based selection for services in cloud environments. In *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pages 268–273. IEEE, 2011.

[21] Alexa the Web Information Company. http://alexa.com/, 2013. [Accessed September 12th, 2013].

[22] Claudorado. Cloud Computing Price Comparison Engine. http://www.cloudorado.com/, 2013. [Accessed September 12th, 2013].

[23] Ciprian Dobre, Florin Pop, and Valentin Cristea. A simulation framework for dependable distributed systems. In *Parallel Processing-Workshops, 2008. ICPP-W'08. International Conference on*, pages 181–187. IEEE, 2008.