Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A SIMHEURISTIC APPROACH FOR RESOURCE ALLOCATION IN VOLUNTEER COMPUTING

Javier Panadero
Laura Calvet
Joan Manuel Marquès
Angel A. Juan

Computer Science Department
Open University of Catalonia - IN3
Av. Carl Friedrich Gauss
Castelldefels, 08860, SPAIN

ABSTRACT

The number of projects relying on volunteer computing and their complexity are growing fast. This distributed paradigm enables the gathering of idle resources (processing power and storage) to run large systems by providing scalable, practical and low cost platforms. The heterogeneity of the resources and their unreliable behavior call for advanced optimization methods. In particular, an efficient resource allocation is key for the systems' performance. This work presents a mathematical formulation and a solving approach based on a metaheuristic for the resource allocation problem. This approach is designed to deal with data-intensive applications, which must guarantee the availability of the data at all times. Moreover, a simheuristic is proposed to deal with the stochasticity of resources' quality. A set of computational experiments are performed to: (1) compare the performance of the metaheuristic and the simheuristic in a stochastic environment; and (2) quantify the effect of the stochasticity on the solutions.

1 INTRODUCTION

Volunteer computing (VC) is becoming increasingly popular because it may provide scalable, practical and low cost platforms. This computing paradigm relies on computational resources (nodes) donated by volunteers. As a consequence, VC systems tend to be large-scale, heterogeneous and distributed. They aim to satisfy computational and storage demands of a number of applications. A disadvantage in comparison with dedicated servers is that these systems may suffer from a lack of reliability, since users are free to connect and disconnect the nodes whenever they want. This behavior may cause significant delays or a loss of data. Thus, it is essential to assign users (*i.e.*, participants of the system who require computational resources) to nodes (see Figure 1) in a smart way to guarantee the fulfillment of the tasks or the availability of the data, and to minimize the quantity of nodes required. In order to create a robust system, replication techniques are used. However, it increases the quantity of nodes needed. A selection mechanism is expected to combine nodes with different availability levels guarantying that the whole system achieves a good quality of service (QoS). Given the dynamism in the VC systems, this mechanism should be fast in order to readjust the map of assignations as soon as possible.

Since the problem is an extension of the facility location problem, which is $\mathscr{NP}$-hard (Mirchandani and Francis 1990), it is also $\mathscr{NP}$-hard. Thus, a heuristic / metaheuristic procedure (Talbi 2009) is required to obtain high quality solutions to real-size instances in reasonable amounts of computing time. We suggest the variable neighborhood search (VNS) metaheuristic (Hansen et al. 2010). In order to take account of the stochasticity associated to the nodes' quality, another solving approach is proposed which assumes that these qualities are random variables that follow specific probability distributions, either theoretical or

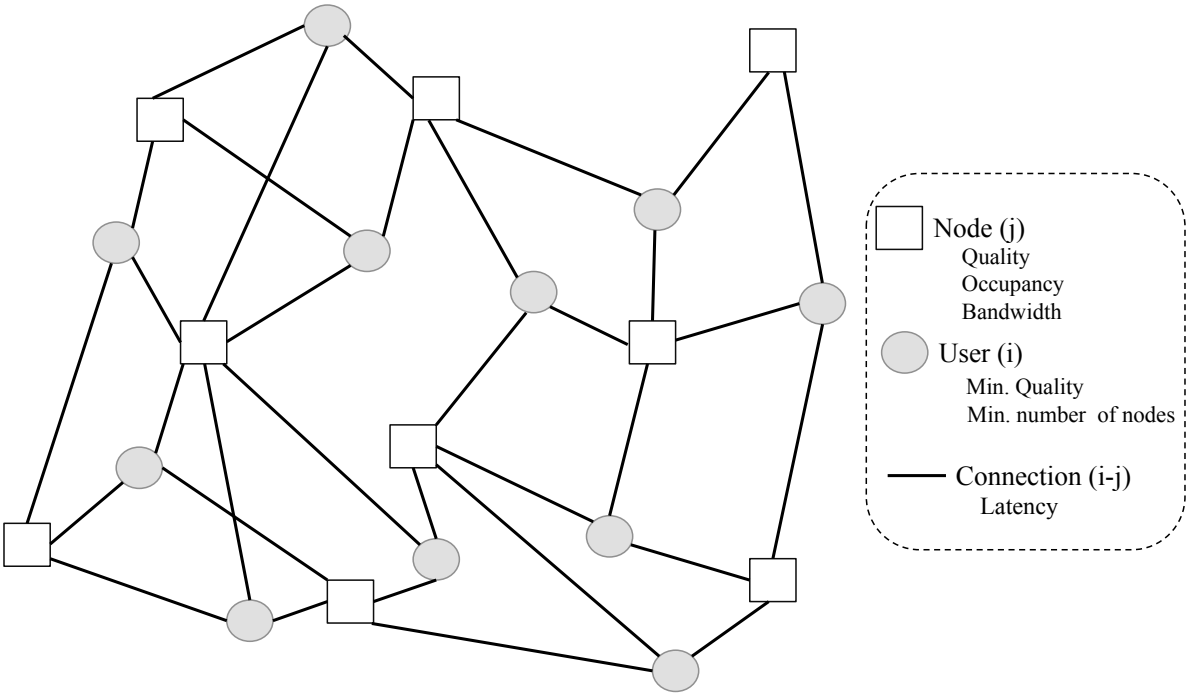

Figure 1: Resource allocation problem.

empirical ones. In particular, a simheuristic (Juan et al. 2015) integrating Monte Carlo simulation (MCS) into the VNS metaheuristic. While the metaheuristic searches for promising solutions, MCS techniques enable the assessment of these solutions in a stochastic environment. This approach is designed to deal with data-intensive applications, which need to guarantee the availability of the data at all times. A set of computational experiments are carried out to illustrate both the problem and the methodologies, and to compare their performance in a number of scenarios.

The rest of this paper is organized as follows. Section 2 reviews some of the existing literature on resource allocation in VC. Section 3 formally introduces the problem, including a mathematical model of the version considered in this work. Section 4 provides a description of our metaheuristic and simheuristic approaches. While section 5 describes an extensive set of computational experiments, section 6 extends the analysis of the results. Finally, Section 7 outlines the main conclusions and provides some suggestions for future research.

2 RELATED WORKS

The literature on the selection of nodes to assign users or tasks in VC systems has increasingly grown during the last decade. For instance, Estrada et al. (2008) design a distributed genetic algorithm to build scheduling policies in VC systems. The method searches over a wide space of scheduling policies generated by employing a subset of IF-THEN-ELSE rules. Guler et al. (2015) present a number of heuristics which distribute jobs aiming to maximize the amount of work done by the users without violating money budget constraints. The heuristics rely on the price of the electricity consumed by the peers and its temporal variation, and the CPU time used. Ghafarian et al. (2013) and Ghafarian and Javadi (2015) develop methods to schedule scientific and data intensive applications workflow, in order to enhance the use of VC systems and increase the percentage of workflow that meets the deadline satisfying QoS constraints. Cabrera et al. (2014) focus on the random behavior of the resources regarding the times they are online (available) and offline. The authors propose a hybrid algorithm combining a metaheuristic with discrete-event simulation.

In the context of resources assignation in online distributed social networks, Duong-Ba et al. (2014) propose several heuristics to address the client-server assignment problem, aiming to find an optimal (or near-optimal) assignment that minimize the total communication load and also achieve a reasonable load balance. User communication patterns are key in the heuristics. Considering the same problem, Zhang and Tang (2014) describe a few heuristics for continuous distributed interactive applications. The heuristics attempt to reduce the network latency to maximize the interactivity under consistency and fairness requirements. Nishida and Nguyen (2011) develop a heuristic based on relaxed convex optimization, which provides a near-optimal client-server assignment for a prespecified trade-off between load balance and communication. A communication pattern represents the input of the heuristic, which can be used in distributed scenarios such as instant messaging systems. Selimi et al. (2016) discuss the challenges of community network micro-clouds such as the dynamic nature of micro-clouds, limited capacity of nodes and links, asymmetric quality of wireless links for services, and deployment models based on geographic singularities rather than network QoS, among others. The authors present a bandwidth-aware service placement algorithm which outperforms the current random placement adopted by *Guifi.net* (a project originally created to solve the broadband Internet access difficulties in rural areas in Catalonia, Spain, with more than 32,500 operating nodes). Panadero et al. (ited) develop the multicriteria biased randomized method to select nodes ensuring a minimum QoS to the users. The method is based on a lexicographic ordering multicriteria strategy based on the intrinsic properties of the donated nodes. In addition, biased randomization techniques are used to distribute and balance the load of the nodes. It is tested by simulating a real large-scale social network.

3 DESCRIPTION OF THE PROBLEM

The (deterministic) resource allocation problem is defined over a set of users $U = \{1,\dots,n\}$ and a set of nodes $R = \{1,\dots,m\}$. Each user i ($\forall i \in U$) requires a deployment with a quality not lesser than q_i and a minimum number of nodes s_i. Similarly, each node j ($\forall j \in R$) has a quality r_j, a bandwidth b_j, a maximum number of connections t_j. For each pair user-node, l_{ij} represents the latency. There is a maximum latency (l_{max}) for all the connections. In addition, a minimum bandwidth (b_{min}) is required. The decision variable x_{ij} is equal to 1 if the user i is assigned to the node j, and 0 otherwise. y_j constitutes another binary decision variable that is equal to 1 when the node j is used, and 0 otherwise.

The problem can be formally described as:

$$\text{Min} \sum_{\forall i \in U} \sum_{j \in R} x_{ij} \tag{1}$$

$$\sum_{\forall j \in R} r_j \cdot x_{ij} \geq q_i \qquad \forall i \in U \tag{2}$$

$$\sum_{\forall j \in R} x_{ij} \geq s_i \qquad \forall i \in U \tag{3}$$

$$b_j \geq b_{min} \cdot y_j \qquad \forall j \in R \tag{4}$$

$$\sum_{\forall i \in U} x_{ij} \leq t_j \qquad \forall j \in R \tag{5}$$

$$l_{ij} \cdot x_{ij} \leq l_{max} \cdot x_{ij} \qquad \forall i \in U, \forall j \in R \tag{6}$$

$$x_{ij} \leq y_j \qquad \forall\ i \in U, \forall j \in R \tag{7}$$

$$y_j \leq \sum_{\forall i \in U} x_{ij} \qquad \forall j \in R \tag{8}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in U, \forall j \in R \tag{9}$$

$$y_j \in \{0,1\} \qquad \forall j \in R \tag{10}$$

The objective function (Equation 1) minimizes the number of connections/assignations established. The restrictions are described next. Equation (2) and Equation (3) impose the number of nodes to which a user is assigned and their quality are greater than or equal to minimum values established. Equation (4) forces to only select nodes with a relatively high bandwidth. Equation (5) ensures that the number of connections for a given node does not excess the maximum. Equation (6) limits the maximum value of the connections' latency. Finally, Equations (7) and (8) relate the decision variables, and Equations (9) and (10) determine their domain.

4 METHODOLOGY

First, this section describes the metaheuristic for the resource allocation problem considering deterministic nodes' quality. Later, the simheuristic that models qualities as random variables is presented.

Metaheuristic algorithm

Our solving methodology is based on the VNS metaheuristic, a popular metaheuristic in both combinatorial and global optimization. It was first proposed by Mladenović and Hansen (1997), and has been successfully applied in a number of research fields such as scheduling, vehicle routing, telecommunications, biology, and artificial intelligence. Hansen et al. (2008), Hansen et al. (2008) and Moreno-Vega and Melián (2008) provide comprehensive reviews of this metaheuristic. Basically, the VNS metaheuristic performs systematic changes of neighborhood in order to find a local minimum by intensifying the search, and to escape from the associated valley by diversifying. The main assumptions are: *(i)* a local minimum with respect to one neighborhood structure is not necessarily so for another; *(ii)* a global minimum is a local minimum with respect to all possible neighborhood structures; and *(iii)* typically, local minimum with respect to one or several neighborhoods are relatively close to each other. The basic version (Algorithm 1) can be described as follows. The number of neighborhoods (K) and the maximum computational time (T) constitute the inputs. In the literature, K is usually set to a small value and the neighborhoods are nested. First, an initial solution (*currentSol*) is built. An outer loop sets the current neighborhood to the first one and includes another loop, which builds and assesses new solutions. These solutions are created by 'shaking' the current solution considering its k-th neighborhood, and improved by means of a local search. If there is an improvement (*i.e.*, *newSol* is preferred over *currentSol*), the new solution is copied into *currentSol* and the current neighborhood is set to the first one. This process represents a descending phase which aims to find a local minimum. Otherwise, the next neighborhood is studied. The inner loop is executed until the last neighborhood is explored. Finally, *currentSol* is returned.

The initial solution is generated as follows. First, the nodes are sorted by quality, from the best to the worst, while the users are randomly ordered. For each user, nodes are iteratively selected until both the minimum quality and the minimum number of nodes required are achieved. Before assigning a node, it has to be checked that the maximum number of connections is not exceeded. Instead of choosing always the nodes in the first positions (*i.e.*, those with the best quality), biased randomization techniques (Juan et al. 2010) are employed to select nodes relying on a Geometric distribution, which boosts the diversification of our algorithm.

The shaking of a given solution destroys the connections of $p\%$ of the users, maintaining all the other connections. The partial solution is repaired by establishing new connections with the same procedure used to generate the initial solution.

After the shaking stage, the local search attempts to improve *newSol* and find a local optimum by applying fast movements. First, the users are decreasingly sorted by their number of connections in decreasing order. For each user with a number of connections greater than the minimum, the following steps are taken: (1) the nodes are sorted by quality in decreasing order; (2) a list of the other users with at least two connections less is created; (3) for each node, the algorithm checks whether the number of connections of that user can be reduced replacing some of them by one to that node considering the option

Algorithm 1 Basic structure of the VNS metaheuristic.

```
    VNS(instance, K, T)
 1: t ← 0
 2: initSol ← initialSolution(instance)
 3: currentSol ← initSol
 4: do
 5:     k ← 1
 6:     while (k ≤ K) do
 7:         newSol ← shake(currentSol,k)
 8:         newSol ← localSearch(newSol)
 9:         if (newSol ≻ currentSol) then
10:             currentSol ← newSol
11:             k ← 1
12:         else
13:             k ← k+1
14:         end if
15:     end while
16:     t ← elapsedTime
17: while (t < T)
18: return currentSol
```

of modifying the connections of one user at most from the list. It is important to highlight that only feasible solutions are considered by the algorithm.

Simheuristic algorithm

This approach, represented in Figure 2, builds on the previous one to account for the stochasticity of the nodes' quality. It is considered that each quality is a random variable following a probability distribution. Basically, the approach integrates MCS into the VNS metaheuristic.

The stochasticity affects the Equation (2) of the model. For a given solution, it cannot be guaranteed that this restriction will not be violated when applied in a stochastic environment. However, the decision-maker may specify a minimum probability associated to the restriction.

The approach includes MCS to estimate the probability for a given solution by following these steps: (1) create a number of scenarios, where each scenario is characterized by a specific value for each random variable generated from the corresponding probability distribution; (2) for each scenario, compute the proportion of users who do achieve the quality required; (3) compute the mean of these proportions, so called reliability of the specific solution. Since MCS techniques tend to be time-consuming, they are only used to assess promising solutions. The solutions labeled as promising are the initial one and those which provide a lower number of connections than *currentSol*. In addition, a relatively small number of scenarios is simulated to obtain estimates in small amounts of time during the main loop. Instead of storing only the best stochastic solution (*i.e.*, that with the lowest reliability), the top best solutions assessed are stored. Once the loop is finished, MCS is applied to those solutions with a higher number of scenarios to obtain more accurate estimates.

5 COMPUTATIONAL EXPERIMENTS

Tests

The metaheuristic algorithm has been implemented using the programming language Java Standard Edition 7.0. Java permits a rapid, platform-independent, development of object-oriented programs, and is popular

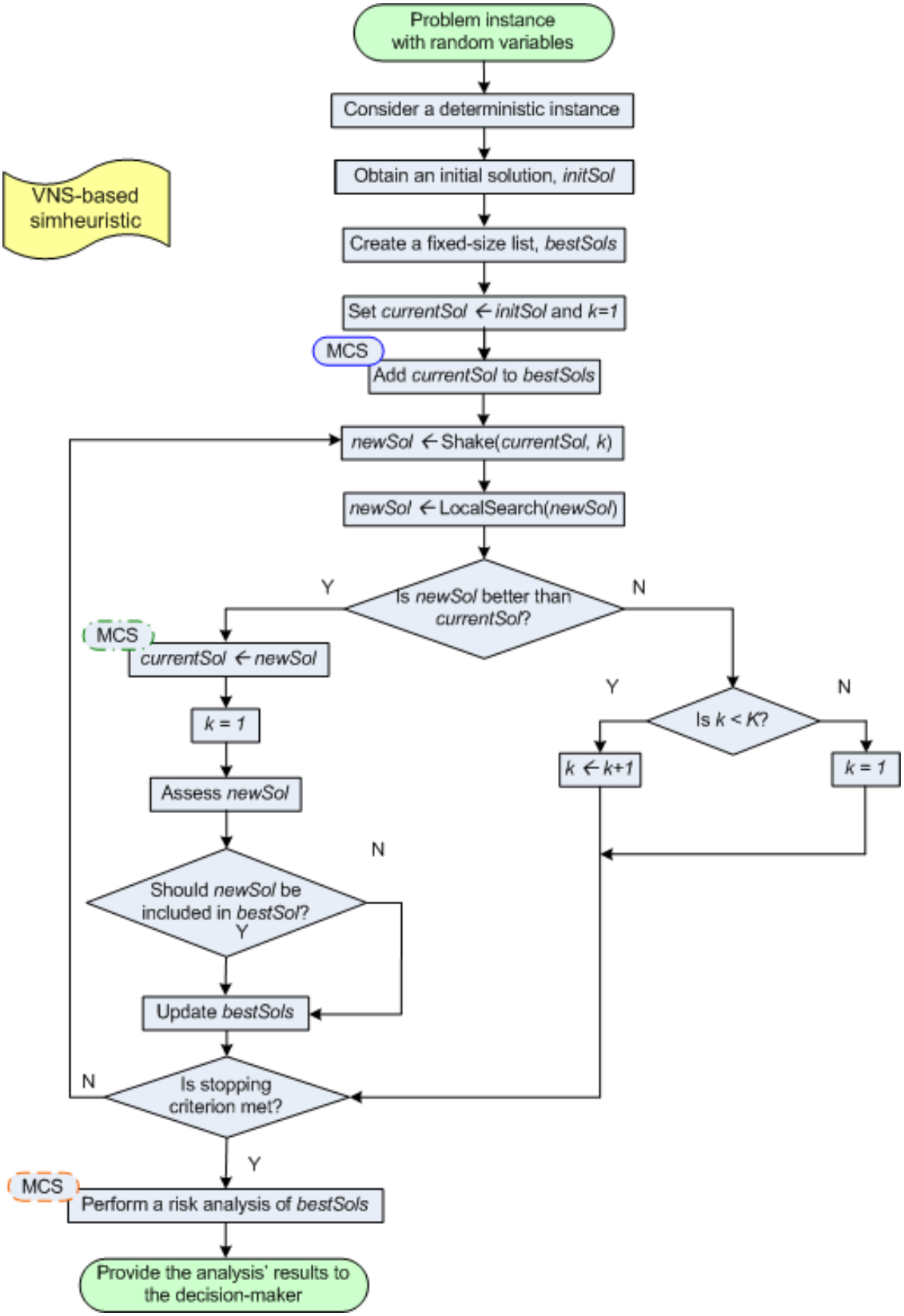

Figure 2: Proposed VNS-based simheuristic for the resource allocation problem.

Table 1: Composition of nodes per instance.

	High	Medium	Low
Instance 1	5%	20%	75%
Instance 2	5%	40%	55%
Instance 3	10%	45%	45%
Instance 4	10%	50%	40%
Instance 5	10%	60%	30%
Instance 6	20%	40%	40%
Instance 7	70%	15%	15%
Instance 8	70%	20%	10%

for developing optimization algorithms. Moreover, its use allows us to integrate the library of stochastic simulation 'ssj' (http://www-labs.iro.umontreal.ca/~simardr/ssj/), which provides the tools to compute multiple measures related to probability distributions, applying quasi-Monte Carlo methods. All the computational experiments have been carried out on a workstation with an AMD quad-core processor of 2.3Ghz with 4GB of RAM memory. As operating system we have used Windows 7.

The algorithm has been executed ten times with different seeds, storing only the best solution. The maximum computing time has been set to 150 seconds per execution. The VNS metaheuristic has 5 neighborhoods, characterized by a given value of p: 5, 10, 15, 20, and 25, respectively. The parameter of the Geometric distribution employed to apply biased randomization is set to 0.5. The number of scenarios simulated to assess the promising solutions is 200, and the number of scenarios to obtain accurate estimates for the best solutions is 2000.

Instances

In order to illustrate and test our approaches with realistic instances, we have considered a simulation of a real micro-blogging application called Garlanet (Garlanet 2016 and Serra et al. 2016). It is a Twitter-like decentralized alternative implementation of a micro-blogging social network. Thus, it constitutes a real-time data-intensive application. It stores the data in heterogeneous computers voluntarily contributed by its participants. In more detail, there is a microservice for each user that keeps its messages and data information. These microservices are replicated across different nodes to guarantee their availability. The application has a centralized control system (CCS), which detects available nodes at any moment and assigns the most suitable nodes to each microservice. Moreover, the CCS guarantees that all the users meet the constraints of minimum number of nodes and minimum QoS. The quality of each user is defined as the sum of the quality of the nodes that host its data.

The simulator defines three kinds of nodes according to their quality: low, middle and high. While nodes of the first type have a high probability of disconnection and a low probability of reconnection, those of the last type show the opposite behavior. Thus, the quality of a node is represented as the predicted probability of being connected during a specific period of time. This is a normalized value between 0 and 1, which is updated as the simulation progresses. To generate diverse instances, we have randomly taken snapshots of the simulator outputs. The generated instances are composed of different percentages of high, medium, and low quality nodes. Table 1 depicts the combinations considered. There are realistic instances (1 to 6) where *low* quality nodes prevail, and other 'ideal' cases (7 and 8), which are more unlikely. The number of nodes and users for all instances are set to 300 and 2100, respectively. In other words, a seventh of the users contribute donating resources. The minimum quality required for users is 2.5.

The instances described are deterministic. Thus, in order to assess the simheuristic approach, they have been adapted by replacing the deterministic quality of the nodes by random variables. In particular, we have employed $TN(\mu,\sigma,l,u)$ referring to the truncated Normal distribution, where the parameters are the mean, the standard deviation, and the lower and upper limits, respectively. We have considered: $\mu = r_j$ and

$\sigma^2 = c \cdot \mu$. 6 values of c (0.0025, 0.01, 0.05, 0.07 and 0.09) have been tested in order to explore different realistic levels of stochasticity.

6 ANALYSIS OF RESULTS

This section presents and discusses the results obtained in two computational experiments. The subsection 6.1 analyzes results for the deterministic version of the problem (*i.e.*, assuming deterministic nodes' quality), while the subsection 6.2 discusses the stochastic version considering that the decision-maker requires a minimum probability of satisfying the constraint of the nodes' quality.

6.1 Deterministic environment

Table 2 gathers the results for the deterministic version. Column 2 represents the number of connections (user-node) needed to meet all the requirements and constraints of the system. Column 3 shows the computational time required to find the best deterministic solution. Columns from 4 to 9 show the reliability of the solution when it is applied in a stochastic environment for different levels of variability. The reliability is defined as the mean proportion of users with a quality higher than or equal to the minimum required, considering all the scenarios. We have used the constraint solver CPLEX to address the last instance, and it has provided the same solution in 76 seconds. Thus, the difference of times (1.24 seconds with the metaheuristic vs 76) is significant.

According to the results, as the percentage of nodes with a low quality increases, the total number of connections also increases. This happens because the users need more replicas to meet the minimum quality requirement. The instances 7 and 8 (which represent the best cases, where the 70% of the nodes have a high quality) have 6300 connections. This number of connections represents the 'ideal' case, where all the users meet the requirements and system constraints, using the minimum number of mandatory replicas per user. Oppositely, the instance 1 (which represents the most difficult case) requires 10324 connections, an average of 5 connections per user.

MCS has been applied to assess the performance of the best deterministic in environments with different levels of uncertainty. Figure 3 depicts the results. The reliability degrades for all the instances as the level of uncertainty increase. The instances from 5 to 8 present a reliability of 100% when the value 0.0025 of c (the lowest uncertainty) is applied. This occurs due to the dominance of high and medium quality nodes, and the low level of variance. For the other instances, the reliability is about 98.74% in average. When the highest level of variance is used ($c = 0.09$), the instance 1 presents the worst reliability, while the best instances (7 and 8) presents a reliability over the 95% due the high number of nodes with high quality. The other instances present a reliability over the 80%.

Table 2: Best deterministic solution in deterministic and stochastic environments

	Best deterministic solution							
			Reliability (%) in a stochastic environment					
	Connections []	Compt. time (s)	c=0.00025 [RS1]	c=0.01 [RS2]	c=0.03 [RS3]	c=0.05 [RS4]	c=0.07 [RS5]	c=0.09 [RS6]
Instance 1	10324	1.86	98.30	94.64	91.48	89.34	84.32	78.32
Instance 2	9573	1.43	99.20	97.54	95.94	92.45	87.53	80.97
Instance 3	9120	1.35	99.12	98.32	97.76	94.95	87.54	81.34
Instance 4	8932	1.75	98.34	97.86	94.97	91.35	86.34	81.34
Instance 5	8134	0.98	100.00	98.54	94.03	92.18	87.33	84.82
Instance 6	7935	1.35	100.00	98.34	97.65	94.54	87.22	82.98
Instance 7	6300	1.32	100.00	100.00	97.34	97.34	96.86	95.56
Instance 8	6300	1.24	100.00	100.00	98.20	98.43	97.28	97.22

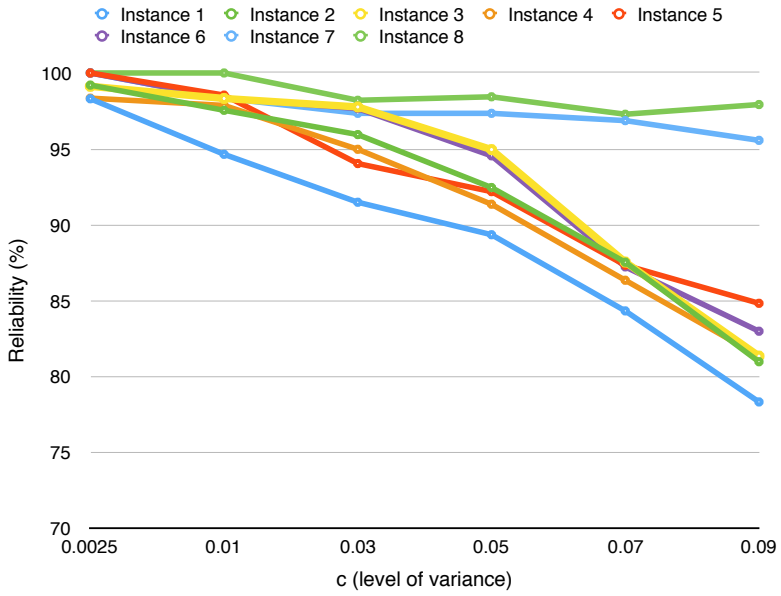

Figure 3: Reliability of the best solutions for different levels of stochasticity.

6.2 Stochastic environment

Table 3 presents the stochastic results using $c = 0.07$ with different levels of minimum reliability required. It can be observed that as the reliability required grows, the number of connections increases to meet this requirement. The average computing time required to find the best stochastic solution was 98 seconds (approximately 84% of the computing time is devoted to the simulation).

Focusing on column 2 (reliability $\geq 82\%$), the number of connections is the same as for the deterministic case. This happens because the system is over this value for all the instances. Concerning the instances 7 and 8, they present the same number of connections for all the levels compared to the deterministic case. The reason is that the reliabilities are greater than 95%, due to the predominance of high quality nodes. Focusing on the instance 5, which presents the most likely scenario in a real system, the effects of increasing the reliability required are less significant than with the other instances. Referring to the instance 1 (the worst scenario), from a reliability higher than 85%, the total number of connections increases to meet the demanded reliability. In this case, the metaheuristic provides a solution with 10624 connections, an average of 5 connections per user.

Table 3: Best stochastic solution with $c = 0.07$ for different levels of reliability.

	Best stochastic solution				
	Reliability ≥ 82	Reliability ≥ 85	Reliability ≥ 90	Reliability ≥ 92	Reliability ≥ 95
Instance 1	10324	10624	12182	12953	15026
Instance 2	9573	9573	10825	11493	12867
Instance 3	9120	9120	9956	10407	12073
Instance 4	8932	8932	10253	11082	12392
Instance 5	8134	8134	8806	9313	9923
Instance 6	7935	7935	8462	8925	10334
Instance 7	6300	6300	6300	6300	6300
Instance 8	6300	6300	6300	6300	6300

7 CONCLUSIONS

Systems relying on volunteer computing are increasing both in number and complexity. Their performance greatly depends on an efficient allocation of resources, which constitutes a challenging task. A mathematical formulation has been provided for this problem. It minimizes the number of connections, considering constraints related to the number of nodes and the quality required by users, and latencies and bandwidth levels. A solving approach based on the VNS metaheuristic has been proposed. However, these systems tend to be very dynamic because of the behavior of the nodes. Accordingly, a simheuristic approach has been described to take account of the stochasticity related to the nodes' quality. It integrates Monte Carlo simulation into the metaheuristic. The idea of this approach is that ignoring the stochasticity may lead to a solution that provides a poor reliability when implemented in a real-life environment. In contrast, the proposed approach allows us to: (1) compute the reliability of solutions; and (2) guarantee a minimum level of reliability. A set of computational experiments have been carried out using realistic instances provided by a simulator of a micro-blogging application. These experiments illustrate the approaches and allow us to compare them in scenarios with different levels of stochasticity.

Several lines of future research stem from this work. For instance, different metaheuristics could be compared, and the stochasticity related to latencies could be considered. Similarly, it would be interesting to study the stochasticity of several real systems and perform sensitivity analysis to understand how the stochasticity affects the systems' performance.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (TRA2013-48180-C3-P, TRA2015-71883-REDT), FEDER, and the Erasmus+ programme (20161ES01KA108023465). We also thank the support of the UOC doctoral program.

REFERENCES

Cabrera, G., A. L. Juan, D. Marques, and I. J. Proskurnia. 2014. "A Simulation-optimization Approach to Deploy Internet Services in Large-scale Systems with User-provided Resources". *Simulation* 90 (6):644-659.

Duong-Ba, T., T. Nguyen, B. Bose, and D. A. Tran. 2014. "Distributed Client-server Assignment for Online Social Network Applications". *IEEE Transactions on Emerging Topics in Computing* 2 (4):422-435.

Estrada, T., O. Fuentes, and M. Taufer. 2008. "A Distributed Evolutionary Method to Design Scheduling Policies for Volunteer Computing". In *Proceedings of the 5th Conference on Computing Frontiers*, edited by A. Ramirez, G. Biliardi and M. Gschwind, 313–322: ACM.

Garlanet 2016. *Garlanet*. Available at http://http://dpcs.uoc.edu/projects/garlanet.

Ghafarian, T., H. Deldari, B. Javadi, M. H. Yaghmaee, and R. Buyya. 2013. "CycloidGrid: A Proximity-aware P2P-based Resource Discovery Architecture in Volunteer Computing Systems". *Future Generation Computer Systems* 29 (6):1583-1595.

Ghafarian, T., and B. Javadi. 2015. "Cloud-aware Data Intensive Workflow Scheduling on Volunteer Computing Systems". *Future Generation Computer Systems* 51:87–97.

Guler, H., B. B. Cambazoglu, and O. Ozkasap. 2015. "Task Allocation in Volunteer Computing Networks under Monetary Budget Constraints". *Peer-to-Peer Networking and Applications* 8 (6):938–951.

Hansen, P., N. Mladenović, and M. J. A.. 2008. "Variable Neighborhood Search". *European Journal of Operational Research* 191 (3):593-595.

Hansen, P., N. Mladenović, and J. A. Moreno. 2008. "Variable Neighborhood Search: Methods and Applications". *4OR - A Quarterly Journal of Operations Research* 6:319–360.

Hansen, P., N. Mladenović, and J. A. Moreno. 2010. "Variable Neighbourhood Search: Methods and Applications". *Annals of Operations Research* 175 (1):367-407.

Juan, A. A., J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. 2015. "A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Combinatorial Optimization Problems". *Operations Research Perspectives* 2:62–72.

Juan, A. A., J. Faulin, J. Jorba, D. Riera, D. Masip, and B. Barrios. 2010. "On the Use of Monte Carlo Simulation, Cache and Splitting Techniques to Improve the Clarke and Wright Savings Heuristics". *Journal of the Operational Research Society* 62:1085–1097.

Mirchandani, P. B., and R. L. Francis. 1990. *Discrete Location Theory*. Wiley.

Mladenović, N., and P. Hansen. 1997. "Variable Neighborhood Search". *Computers & Operations Research* 24 (11):1097–1100.

Moreno-Vega, J. M., and B. Melián. 2008. "Introduction to the Special Issue on Variable Neighborhood Search". *Journal of Heuristics* 14 (5):403-404.

Nishida, H., and T. Nguyen. 2011. "Optimal Client-Server Assignment for Internet Distributed Systems". In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, edited by D. Bader, 1–6: IEEE Press.

Panadero, J., J. de Armas, X. Serra, and J. Marquès. submited. "Multi Criteria Biased Randomized Method for Resource Allocation in Distributed Systems: Application in a Volunteer Computing System". *Future Generation Computer Systems*.

Selimi, M., L. Cerdà-Alabern, L. Wang, A. Sathiaseelan, L. Veiga, and F. Freitag. 2016. "Bandwidth-Aware Service Placement in Community Network Micro-Clouds". In *41st Conference on Local Computer Networks (LCN)*, edited by S. Kanhere and A. B. Mnaouer, 220–223: IEEE.

Serra, X., J. de Armas, and J. M. Marquès. 2016. "Simulating and optimizing resource allocation in a micro-blogging applicacion". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. Roeder, P. I. Frazier, R. Szechtman and E. Zhou, 3167–3176. Piscataway, NJ, USA: IEEE Press.

Talbi, E. G. 2009. *Metaheuristics: From Design to Implementation*. Wiley.

Zhang, L., and X. Tang. 2014. "The Client Assignment Problem for Continuous Distributed Interactive Applications: Analysis, Algorithms, and Evaluation". *IEEE Trans. Parallel Distrib. Syst.* 25 (3):785–795.

AUTHOR BIOGRAPHIES

JAVIER PANADERO is a PostDoc researcher at the Computer Science, Multimedia and Telecommunication Department at Open University of Catalonia (UOC). His major research areas are: performance prediction of HPC applications, Modeling and analysis of parallel applications, Simulation and metaheuristics. He has co-authored a total of 11 full-reviewed technical papers in journals and conference procedings. His e-mail address is jpanaderom@uoc.edu.

LAURA CALVET is a PhD student at the IN3 - Open University of Catalonia. She holds a MSc in Statistics and Operations Research, one BSc in Economics and another in Applied Statistics. Her work is related to the combination of statistical learning and simheuristics for solving complex combinatorial optimization problems under uncertainty. Her email address is lcalvetl@uoc.edu.

JOAN MANUEL MARQUÈS is an Associate Professor at Computer Sciences, Multimedia & Telecommunication Studies at Universitat Oberta de Catalunya (UOC) since 1997. He graduated as a Computer Science Engineer from the Facultat d'Informàtica de Barcelona (UPC) in 1991 and received his PhD from UPC in 2003. His research interests include the design of scalable and cooperative Internet services and applications. He is member of the Association for Computing Machinery. His email address is jmarquesp@uoc.edu.

ANGEL A. JUAN is Associate Professor in the Computer Science Dept. at the Open University of Catalonia, and Principal Investigator at IN3. He holds a PhD in Industrial Engineering and a MSc in Mathematics. His research interests include applications of randomized algorithms and simheuristics in

logistics, transportation, and finance. He has published over 50 JCR-indexed articles in these fields. His website address is http://ajuanp.wordpress.com and his email address is ajuanp@uoc.edu.