An Analysis of Optimization Algorithms Designed to Fully Comply with SLA in Cloud Computing

L. H. V. Nakamura, L. J. M. de Azevedo, J. C. Estrella, C. F. M. Toledo, B. G. Batista, R. I. Meneguette, S. R. Marganiec, R. H. C. Santana and M. J. Santana

***Abstract*— Nowadays the access to a cloud computing environment is provided on-demand offering transparent services to customers. Although the cloud allows an abstraction of the behavior of the service providers in the infrastructure (involving logical and physical resources), it remains a challenge to fully comply with the Service Level Agreements (SLAs), because, depending on the service demand and system configuration, the providers may not be able to meet the requirements of the customers. There is a need for mechanisms that take account of load balancing algorithms to provide an efficient load distribution with the available resources. However, the studies in the literature do not effectively address the problem of the availability of resources to meet customers' requirements with analysis restricted to a limited set of objectives. This paper proposes algorithms to address the need for optimization when handling computational resources during the execution time. The methods optimize the efficient use of the resources available in the infrastructure aiming to comply with the service level agreements defined between client and provider.**

***Keywords*— Cloud Computing, Optimization, Genetic Algorithm, SLA, QoS.**

I. INTRODUÇÃO

NOS últimos anos um dos tópicos mais discutidos na área de Tecnologia de Informação (TI) tem sido computação em nuvem. Segundo o Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST - *National Institute of Standards and Technology*) o termo "Computação em Nuvem" é definido da seguinte forma: *"A computação em nuvem é um modelo que permite o acesso a rede de forma ubíqua, conveniente e sob demanda, a um conjunto compartilhado de recursos computacionais configuráveis (tais como, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gestão ou interação com o provedor de serviços"* [1].

A computação em nuvem também pode ser considerada uma extensão de outros paradigmas, como os da computação em grade e utilitária, acrescentando a capacidade de que as aplicações de negócios podem ser expostas como serviços sofisticados e acessados por meio de uma rede (Internet) [2]. Dessa forma, a Computação em Nuvem envolve dois fatores essenciais que são: Computação e Negócios. A computação é fornecida por meio de tecnologias atuais de *software* e *hardware* escaláveis, sob demanda e suportados pelas atuais técnicas de virtualização. Tais recursos devem ser providos de forma adequada e de acordo com os modelos de Negócio estabelecidos entre os clientes e provedores da Nuvem.

A Computação em Nuvem estão conceitualmente dividida em três modelos de serviços básicos que são: *Software as a Service* (SaaS), *Platform as a Service (PaaS)*, e *Infrastructure as a Service (IaaS)*. Esses modelos propõem as capacidades ou as formas de como os serviços na nuvem são prestados aos seus clientes. Essa abordagem pode caracterizar uma divisão em camadas abstrata dos serviços prestados [3].

O SaaS trata da capacidade do provedor em fornecer, ao cliente, aplicativos em execução em uma infraestrutura em nuvem [1]. As aplicações são fornecidas como um serviço e são acessíveis a partir de diversos dispositivos clientes, por meio de interfaces *thin client*, como um navegador web. Alguns exemplos de aplicações incluem, em nível empresarial, a *Salesforce*, *Google Apps* para aplicações pessoais, *Gmail*, *Facebook* e *Twitter* [4].

O PaaS corresponde a capacidade do provedor em fornecer ao cliente a possibilidade de desenvolver aplicações na infraestrutura da nuvem que são criadas usando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. Na visão da Intel, a PaaS fornece uma plataforma para os desenvolvedores que facilita a implementação e implantação de aplicativos (*Web Applications* ou SaaS) evitando o custo e a complexidade da compra e também o gerenciamento de *hardware* e camadas de *softwares* [5]. Pode-se citar dois exemplos mais conhecidos de PaaS como o *Google App Engine* (GAE) e a *Windows Azure* da Microsoft.

A IaaS fornece ao cliente a capacidade de provisionamento de processamento, armazenamento, redes e outros recursos computacionais fundamentais. O cliente é capaz de implantar e executar *softwares* arbitrários, os quais podem incluir sistemas

L. H. V. Nakamura, Universidade de São Paulo (USP), nakamura@icmc.usp.br
L. J. M. de Azevedo, Universidade de São Paulo (USP), leonildo.azevedo@usp.br
B. G. Batista, Universidade Federal de Itajubá (UNIFEI), brunoguazzelli@unifei.edu.br
R. I. Meneguette, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP-Catanduva), meneguette@ifsp.edu.br
C. F. M. Toledo, Universidade de São Paulo (USP), claudio@icmc.usp.br
J. C. Estrella, Universidade de São Paulo (USP), jcezar@icmc.usp.br
S. R. Marganiec, University of Leicester, srm13@le.ac.uk
R. H. C. Santana, Universidade de São Paulo (USP), rcs@icmc.usp.br
M. J. SantanaUniversidade de São Paulo (USP), mjs@icmc.usp.br

operacionais e aplicativos. O cliente não gerencia ou controla a infraestrutura, mas tem controle sobre sistemas operacionais, armazenamento, aplicativos implantados e um eventual e limitado controle dos componentes de rede selecionados (por exemplo, *firewalls* do *host*). Provedores de IaaS normalmente oferecem uma infraestrutura virtualizada como serviço ao invés do *hardware* real diretamente. Tais serviços podem ser fornecidos por interfaces padronizadas para serviços do tipo PaaS e SaaS. Alguns exemplos sao: *Amazon Web Services* (AWS) com os seus serviços de processamento (*Elastic Compute Cloud* EC2) e armazenamento (*Simple Storage Service* – S3) [6].

Dessa forma, pode-se perceber que a Computação em Nuvem e um paradigma de computação vasto e complexo que está intimamente relacionada aos negócios de seus clientes. Para este artigo, o nosso foco está no modelo de serviço IaaS e, dessa forma, propomos a criação e avaliação de algoritmos de otimização para o cumprimento de acordos de níveis de serviços (SLA - *Service Level Agreement*) em Nuvem.

Prover uma infraestrutura adequada aos seus clientes sempre foi um desafio para os provedores de serviços. Clientes diferentes exigem quantidades e recursos diferentes dependendo de suas aplicações hospedadas ou da demanda exigida pelos usuários de tais aplicações. Por exemplo, um determinado˜ cliente pode possuir uma aplicação *CPU-Bound* enquanto outro cliente possui uma aplicação *Memory-Bound*. Um cliente pode ter uma aplicação com uma demanda de acessos simultâneos extremamente alta enquanto outro cliente pode ter uma aplicação com poucos acessos aleatórios. Por esse motivo, os provedores em Nuvem geralmente disponibilizam diversas configurações de recursos virtualizados como, por exemplo, CPU, Memória e Disco, agrupados ou baseados em Instâncias ou Máquinas Virtuais (VMs). Algumas configurações de máquinas virtuais são pré-definidas (por exemplo, a Amazon trabalha com diferentes classes de VMs) pelo provedor em Nuvem e o próprio usuário deve determinar a melhor quantidade e capacidade das máquinas virtuais para que o seu sistema funcione adequadamente.

Uma vantagem da Computação em Nuvem é que os provedores fornecem serviços de monitoramento que permitem a elasticidade (aumentar ou diminuir) da capacidade computacional. Isso é feito baseado na configuração de parâmetros e políticas simples de escalonamento. Dessa forma, sob uma alta demanda de serviços, recursos computacionais podem ser adicionados automaticamente para atender o aumento de requisições. Por outro lado, no caso de baixa demanda, recursos podem ser desalocados automaticamente para a economia do cliente que geralmente paga a utilização dos recursos por hora. No entanto, a grande desvantagem é que tais sistemas de escalonamento dinâmico, fornecidos pelos provedores, consideram basicamente a demanda ou a taxa de utilização dos recursos para realizar o acréscimo ou decréscimo de recursos. Acertar o limiar de quando uma reconfiguração da infraestrutura deve ser realizada e um desafio. Outro ponto interessante é quantificar o volume de recursos (quantidade de instâncias de máquinas virtuais) que devem ser iniciados ou desligados. Portanto, realizar tal reconfiguração da infraestrutura dinamicamente, não apenas com o objetivo de atender a demanda, mas realizar tal procedimento de forma eficiente e almejando outros objetivos como redução dos custos e a garantia da qualidade de serviço definida no SLA estabelecido entre provedor e clientes são tarefas não triviais e que necessitam de estudos adicionais.

Problemas dessa natureza, como escalonamento de tarefas e provisionamento de recursos, sao considerados problemas da classe NP-difícil [7]. Muitos problemas da classe NP-difícil sao resolvidos através da programação inteira é *branch-and-bound* [8], no entanto, não são adequados para resolver problemas de decisão que possuem adaptação contínua [9]. Neste contexto, optamos por investigar algoritmos de otimização encontrados na computação evolutiva, pois apresentam um bom desempenho na resolução de problemas reais.

A computação evolutiva apresenta um bom desempenho na resolução de problemas reais. Segundo [10], há três principais motivações para a utilização dessa abordagem: (1) os métodos de solução desenvolvidos são inspirados em problemas e soluções encontrados na natureza, incluindo os problemas mais complexos, como o cérebro humano e o processo evolucionário; (2) outra motivação vem de uma perspectiva técnica, pois os algoritmos evolutivos retornam uma boa solução (não necessariamente a ótima) com um tempo aceitável; (3) uma terceira motivação é encontrado em um contexto científico, ”a curiosidade humana”, pois o processo evolucionário e objeto de muitos estudos, principalmente para entender como esse processo funciona.

Os Algoritmos evolutivos são amplamente utilizados em problemas de otimização, principalmente aqueles problemas intratáveis, onde o tempo necessário para resolvê-lo e inaceitável. Em outras palavras, pode-se dizer que um problema é intratável, se o seu limite de complexidade é exponencial como n! ou k^n, sendo k uma constante qualquer maior que 1 [11].

Com o objetivo de sanar os desafios destacados foi modelado o problema (descrito na seção III) e desenvolvidos algoritmos de otimização que também passaram por uma avaliação. Dentre eles, foram desenvolvidos um algoritmo determinístico e um micro Algoritmo Genético (μGA). A implementação dos algoritmos e os resultados obtidos são apresentados no decorrer deste artigo.

O restante deste artigo está estruturado da seguinte forma: na Seção II é apresentada uma revisão da literatura que aborda o tema otimização dentro da computação em nuvem. Na Seção III e definido o problema que será abordado e solucionado por este trabalho. A Seção IV descreve os métodos utilizados na solução do problema e as suas aplicações nos algoritmos. Na Seção V estão descritos os resultados das análises de testes computacionais. Finalmente, na Seção VI são descritas as conclusões e as diretrizes para trabalhos futuros.

II. TRABALHOS RELACIONADOS

A nuvem e um ambiente escalável, no qual a demanda de serviços pode mudar instantaneamente. Dessa forma, a alocação automática de recursos para atender essa demanda torna-se um tópico interessante tanto no âmbito acadêmico quanto no industrial. O correto provisionamento permite um melhor uso dos recursos computacionais disponíveis e, consequentemente, de toda a infraestrutura que compõe a nuvem, pois o mapeamento entre a carga e os recursos é mais eficiente. Além disso, isso ajuda no cumprimento das exigências feitas pelos clientes e provê um grande dinamismo ao sistema [12]. A tarefa de prover mais recursos computacionais a um cliente, por exemplo, é altamente viável e de fácil disponibilização, uma vez que os recursos computacionais são virtualizados e podem ser considerados ilimitados na visão dos clientes. No entanto, é importante lembrar que a alocação de mais recursos computacionais influência no custo final que é repassado ao cliente e requer mecanismos eficientes por parte do provedor [13].

Há na literatura diversos trabalhos que analisam e propõem mecanismos para o gerenciamento de recursos em um ambiente em nuvem. A proposta da Amazon para a reconfiguração automática da infraestrutura de seus clientes é baseada em monitoramento, por meio de alertas (*CloudWatch Alarms*) e políticas (*Scaling Policies*). O escalonamento da AWS (*Amazon Web Services*) pode ser feito baseado em métricas que são geralmente baseadas no consumo de CPU [14] ou em políticas que sao basicamente divididas em políticas manuais (*Manual Scaling*), políticas dinâmicas (*Dynamic Scaling*) e políticas agendadas (*Scheduled Scaling*) [15].

As políticas dinâmicas são as mais interessantes para este trabalho. A Amazon permite ao usuário configurar o tipo de escalonamento de recursos baseado no ajuste do número de instâncias que devem ser alocadas ou desalocadas que seguem uma quantidade ou uma porcentagem definida pelo usuário [16]. Dessa forma, fica a cargo do usuário tentar prever qual seria a melhor configuração para um escalonamento eficiente para a sua infraestrutura hospedada na nuvem do provedor. Contudo, nenhuma configuração eficiente é feita para que seja garantida a QoS da infraestrutura.

Em [17], e proposto um "balanceador de carga central" para um ambiente de computação em nuvem de grande escala. Todas as requisições chegam a um centro de controle, no qual o balanceador de carga central e conectado a todos os usuários e a todas as máquinas virtuais presentes nessa nuvem. A prioridade para cada máquina virtual é calculada com base na velocidade da CPU (MIPS) e memória. Com esse escalonador centralizado, foi possível obter um escalonamento aceitável. Porém, diagnosticou-se a necessidade da implementação de outros algoritmos para distribuir a carga de acordo com a utilização dos recursos, de modo a obter uma distribuição de carga mais dinâmica e robusta.

Os autores em [18] investigam diferentes tipos de carga de trabalho para as VMs, a fim de encontrar melhores estratégias de alocação de recursos. Para tal, é levado em consideração um número de parâmetros (críticos) sobre o desempenho das VMs, os quais incluem percentuais de alocação, decisões de escalonamento em tempo real e realocação de VMs. Além disso, é inserido um método "caixa preta" baseado em redes neurais, o qual é comparado com um método de regressão linear. A aplicação de redes neurais para a alocação de recursos demonstrou-se eficiente, apresentando uma margem de erro inferior a 5%.

Dessa forma, o provedor pode ter conhecimento prévio de possíveis interferências na carga de trabalho e, consequentemente, otimizar a alocação de recursos. Por outro lado, são necessárias algumas alterações para que haja a detecção automática do tipo de carga de trabalho gerada por cada aplicação. Além disso, os testes deveriam ser adequados para serem executados em aplicações reais.

Zhao e colaboradores [19] propõem um modelo de escalonamento de recursos virtuais baseado na seleção por meio de algoritmos genéticos multi-objetivo (NSGA II - *Non-dominated Sorting Genetic Algorithm*). Esse modelo foi avaliado pelo equilíbrio da carga distribuída, recursos físicos e virtuais com a análise de escalonamento dos recursos virtuais. Os autores compararam esse modelo com algoritmos de geração aleatório, algoritmos estatísticos e algoritmos de ranqueamento. Com base nos testes executados, esse trabalho mostrou que o NSGA II pode ser utilizado para resolver o problema de escalonamento de recursos virtuais. Nos experimentos nos quais os algoritmos são comparados, o NSGA II demonstrou-se mais apropriado para esse tipo de aplicação. Contudo, esse modelo não leva em consideração as falhas encontradas em um ambiente real, as necessidades do cliente em contraste com o SLA e os recursos disponíveis, bem como o custo cobrado do cliente.

Wu e colaboradores [20] propõem um algoritmo de controle de admissão e escalonamento para que os provedores de SaaS possam utilizar de maneira eficaz os recursos e maximizar seus lucros, além de reduzir os custos e melhorar a satisfação dos clientes. Nesse trabalho, foram apresentadas quatro estratégias para avaliar se uma requisição pode ou não ser atendida de maneira a garantir a QoS estipulada. Na fase de avaliação, os desempenhos dos algoritmos de cada estratégia foram comparados com outros dois de referência na literatura: o *MinResTime* e o *StaticGreedy*. Nas simulações executadas, o algoritmo proposto por esse trabalho conseguiu obter uma redução do custo de até 40% levando em consideração todas as faixas de variação de parâmetros de QoS estipuladas. Apesar dos resultados promissores, diagnosticou-se a necessidade de melhorar a robustez dos algoritmos a fim de tratar erros dinamicamente, bem como a necessidade de considerar negociações do SLA para melhorar o desempenho.

Em outra proposta [21] são descritos os recursos necessários para tornar autônomo um determinado recurso na Nuvem visando manter a QoS com base em alertas de violação, quando o sistema de planejamento recebe esses alertas determina que alguma reconfiguração seja realizada. Assim esses autores propõem gerenciar os processos de negócios baseados em Serviços (*Service-based Business Processes* - SBPs) de forma

elástica.

Outros autores afirmam que a alocação de recursos computacionais para serviços de forma eficiente e um desafio existente no ambiente de computação em nuvem [22]. A solução desse problema proposta por autores é a utilização um corretor de nuvens (*Cloud Broker*) para o gerenciamento de desempenho em nuvem federada. Essa solução também prever a manutenção dos acordos de níveis de serviços (SLAs) por meio da alocação dinâmica de recursos visando o lucro.

Em um trabalho prévio [12], os autores fizeram uma proposta de análise de desempenho no gerenciamento de recursos em computação em nuvem, onde foi implementado um módulo de reconfiguração que tinha como objetivo reduzir o número de violações no SLA. Contudo, constatou-se a necessidade de técnicas de otimização, pois em alguns casos o módulo de reconfiguração não conseguia manter a QoS contratado entre o provedor e cliente. Esse trabalho motivou a proposta deste artigo.

A Tabela II apresenta as principais características dos trabalhos analisados, levando em consideração o SLA, o auto-gerenciamento do mecanismo proposto, a heurística utilizada e a garantia de QoS.

Há diversos problemas complexos dentro do contexto de computação em nuvem que são abordados por soluções que utilizam otimização. Por exemplo, os problemas de alocação de máquinas virtuais em máquinas reais [23], economia de energia [24], escalonamento e balanceamento de carga das aplicações em máquinas virtuais [25] e utilização de recursos visando minimizar o custo e ainda garantir o desempenho [26], a QoS e o cumprimento do SLA.

Todos esses problemas exigem soluções ótimas ou sub-ótimas que podem ser alcançadas por técnicas conceituadas da área de otimização desde que sejam adequadamente mapeadas para solucionar tais problemas práticos que surgiram com o paradigma de computação nas “nuvens”.

Este trabalho visa solucionar o problema de alocação de recursos de forma adequada, utilizando uma heurística e uma metaheurística para alcançar resultados que garantam o SLA, o autogerenciamento e a QoS para os clientes de um provedor em nuvem.

TABELA I
COMPARAÇÃO COM OUTROS TRABALHOS RELACIONADO

Trabalhos Relacionados	**SLA**	**Auto Gerenciamento**	**Heurítica**	**QoS**
Soni and Kalra, 2014 [17]	X	√	X	X
Su et al., 2013 [27]	X	√	Pareto	X
Kousiouris et al., 2011 [18]	X	√	Rede Neural	√
Zhao et al., 2011 [19]	X	√	NSGA II	X
Wu et al., 2012 [20]	√	X	X	√
Mecanismo Proposto	√	√	*µGA*	√

III. FORMULAÇÃO DO PROBLEMA

Em um ambiente em nuvem, quando um mecanismo de provisionamento de recursos bem elaborado é utilizado, as aplicações podem operar mais eficientemente, com redução nos custos, melhor utilização da infraestrutura disponível e melhor desempenho em momentos de pico com variações na demanda de serviços. No entanto, o processo de provisionamento é complexo [28]. De acordo com [29], isto requer a definição das melhores configurações de *software* e *hardware* para garantir que os SLAs sejam cumpridos, além de maximizar a eficiência e a utilização do sistema.

Há diversos desafios relacionados ao provisionamento de recursos e ao atendimento das requisições utilizando a infraestrutura de uma nuvem. Esses desafios integram a modelagem de carga, virtualização, modelagem de desempenho, depuração e monitoramento de aplicações nos recursos virtualizados [30]. Além disso, há situações imprevisíveis que podem prejudicar o eficiente provisionamento e atendimento das requisições durante o tempo de execução, dentre as quais podem ser citadas [31]:

- Erro de estimativa: a combinação errada entre os recursos computacionais e aplicações pode levar a uma sub ou superestimativa na demanda de clientes, o que exerce um grande impacto na QoS contratada e no custo do serviço;
- Carga dinâmica: a nuvem e um ambiente com diferentes tipos de clientes que requisitam diferentes tipos de serviços. Por isso, diferentes picos de carga podem ocorrer, dependendo do dia e da época do ano, ou da popularidade de uma aplicação. Esses fatores são responsáveis por sérios problemas durante a estimativa do comportamento da carga e na definição dos recursos;
- Comportamento inesperado: disponibilidade, carga, vazão (*throughput*) de recursos e conexões da rede podem variar de maneira imprevisível em ambientes de computação em grande escala como em uma nuvem. Essas instabilidades do sistema prejudicam a determinação dos recursos de forma eficiente durante o provisionamento.

Provedores como Amazon EC2 e Microsoft Azure utilizam uma metodologia de provisionamento de recursos na qual os clientes são responsáveis por estimar com precisão a quantidade de recursos necessários e selecionar a instância a ser contratada [32]. Importante lembrar que nem sempre o usuário tem conhecimentos técnicos para fazer um gerenciamento manual do provisionamento de recursos, sem falar que tal tarefa pode ser exaustiva para o usuário. Dessa forma, a aplicação de alguma técnica de otimização para o provisionamento de recurso tornaria tal tarefa automatizada, garantia a máxima utilização dos recursos computacionais e consequentemente, o usuário não pagaria além do que realmente necessita, ou seja, pagaria um preço justo pelo serviço contratado.

Neste contexto, tratamos o problema com VMs heterogêneas baseadas nas instâncias *m3.medium, m3.large* e *m3.xlarge* da Amazon EC2, considerando o SLA para tomada de decisão em tempo real. O SLA e tratado a princípio com quatro atributos de QoS neste artigo, sendo eles:

- **Capacidade (C)**: número de máquinas virtuais contratadas pelo cliente. Podemos considerar inicialmente que há três tipos de *Virtual Machines* (VMs), sendo elas; *Small*, *Medium* e *Large*, tais

instâncias foram baseadas nas configurações das instâncias *m3.medium* e *m3.large* da Amazon EC2. Uma infraestrutura de um *data center* pode ter inúmeras VMs com as mais diversas configurações. Para obter a capacidade foi elaborada a Equação 1, na qual *n* corresponde ao número de máquinas virtuais:´

$$C = \sum_{i=1}^{i=n} \quad Capacidade(VM_i) \tag{1}$$

- **Tempo de Resposta (TR):** o tempo de resposta da aplicação esperado pelo cliente, que e obtido pela execução da aplicação na infraestrutura contratada. O tempo de resposta pode ser obtido pela Equação 2 [33], na qual *n* corresponde ao número de máquinas virtuais:

$$TR = \frac{\sum_{i=1}^{i=n} \quad Tempo\ de\ Resposta(VMi)}{n} \tag{2}$$

- **Disponibilidade (D):** consiste na infraestrutura estar presente e pronta para a utilização. Nesse contexto, a disponibilidade pode ser obtida de duas formas, (i) inicialmente, pode ser calculada pela razão do número de máquinas em relação a probabilidade de ocorrer um erro, ou seja, 1/(número de VMs). Nesse caso se houvesse quatro VMs instanciadas, a probabilidade de ocorrer erro seria de 1/4 (25%) e a disponibilidade seria de 75%; ou ainda, (ii) utiliza-se uma variável contador para calcular o percentual de requisições atendidas com êxito e o percentual de falhas, assim o percentual de requisições atendidas com êxito será o percentual de disponibilidade. Neste trabalho a disponibilidade será calculada das duas maneiras; primeiramente pelo método (i) durante o processo de monitoramento e posteriormente através do método (ii). Para o segundo método, a disponibilidade pode ser obtida pela Equação 3 [33], na qual *n* corresponde ao número de máquinas virtuais:

$$D = \prod_{i=1}^{i=n} \quad Disponibilidade(VMi) \tag{3}$$

- **Custo por hora (C/h):** valor monetário estipulado no SLA, referente a quanto o cliente irá pagar por hora pelo serviço durante a utilização da VM. O custo monetário por hora pode ser obtido pela Equação 4 [33], na qual *n* corresponde ao número de máquinas virtuais:

$$C/h = \sum_{i=1}^{i=n} \quad Custo\ (VM_i) \tag{4}$$

As máquinas virtuais são classificadas de acordo com suas configurações de memória, núcleos virtuais (vCPUs - *virtual CPUs*), e tamanho de disco, e o preço é definido baseado na classe, onde as instâncias mais potentes são as mais caras. É importante observar que, o aumento da quantidade de recursos não implica necessariamente em um melhor desempenho do sistema. No entanto, esse aumento tem um grande impacto sobre o custo. Considere uma situação na qual uma aplicação necessita de mais poder de processamento e nenhuma adição na quantidade de memória, e a próxima classe de VM fornece um aumento nos dois recursos (CPU e memória). O cliente pode alterar de um tipo de VM para outro com mais recursos, pagar mais e, consequentemente, obter um desempenho melhor, mas o aumento do desempenho não está associado ao aumento de todos os recursos que compõem a VM.

Além disso, a Amazon EC2 utiliza um mecanismo chamado *Auto Scalling* que monitora as condições do sistema e altera o número de instâncias de máquinas virtuais, caso seja necessário. Ele adiciona ou remove instâncias, sem alterar as configurações dos recursos que compõem a VM. Na Microsoft Azure, os clientes podem definir limites a partir dos quais a escalabilidade horizontal e aplicada. Essa pode ser aplicada baseada no percentual médio de uso da CPU ou baseada no número de mensagens em uma fila. Se o percentual médio de uso da CPU ou o número de mensagens na fila estão abaixo ou acima dos limites especificados, novas instâncias são criadas ou excluídas, ou máquinas virtuais são ligadas ou desligadas a partir de um conjunto de máquinas criadas previamente.

Dessa forma, os provedores citados não alteram as configurações das VMs durante o tempo de execução. Eles alteram o número de instâncias respeitando as classes definidas por eles, onde eles adicionam ou removem VMs da mesma classe (os provedores chamam esse procedimento de escalabilidade horizontal) ou eles adicionam ou removem VMs de diferentes classes, por exemplo, um cliente contrata uma VM do tipo *Small* e por alguma razão uma nova VM do tipo *Large* e ligada (os provedores chamam esse procedimento de escalabilidade vertical). Dessa forma, é mais fácil para os provedores como a Amazon definir instâncias com configurações fixas ao invés de criar uma instância com configurações específicas para cada cliente.

No trabalho apresentado nesse artigo acreditamos que o provisionamento de recursos deve ser realizado automaticamente e dinamicamente baseado nas exigências feitas pelos clientes no SLA, a um preço justo e na demanda de serviços.

IV. MÉTODOS

Uma vez definidos os parâmetros de QoS (Seção III), alguns experimentos foram realizados no simulador CloudSim versão 3.0.3, utilizando um computador com um processador AMD Phenom(tm) II X6 1090T, com 16 GB de memória RAM, com 1,5 TB´ de armazenamento em disco e sistema operacional Ubuntu 14.04.3 LTS. O CloudSim foi configurado com os tres tipos^ de instâncias descritos na Seção III.

Na execução dos experimentos foi estipulado um SLA para um cliente, tendo como atributos de QoS: Capacidade (C^c), o Tempo (T^c), Disponibilidade (A^c) e Custo/h (C/h^c). A simulação consiste em executar a aplicação do cliente na capacidade descrita no SLA. Com a execução da simulação é obtido o tempo de resposta, a disponibilidade e o custo/h. Caso os parâmetros retornados pelo simulador não sejam compatíveis com os descritos no SLA, posteriormente é aplicada uma metaheurística para encontrar uma solução (capacidade (C^*), tempo (T^*), disponibilidade (A^*) de custo/h (C/h^*)) que melhor se adeque às exigências do cliente. No entanto, é difícil

satisfazer todas essas exigências sem conflitos. Por exemplo, C^c pode não ser satisfeito com o custo desejado C/h^c ou não pode executar uma aplicação dentro do tempo T^c.

Assim, foram elaborados dois algoritmos para a solução do problema, um algoritmo determinístico (Fig. 1) e um μAG (Fig. 2). Esses métodos otimizar ao o número de máquinas virtuais contratadas pelo cliente, conforme descrito na Seção III. Assim, a representação da solução (codificação) é definida por (*s,m,l*), que são o número de máquinas virtuais (VMs) do tipo *Small*, *Medium* e *Large*, respectivamente, que melhor satisfazem as solicitações dos clientes. Se esses valores desses parâmetros não são compatíveis com aqueles definidos no SLA (C^c, T^c, A^c, C/h^c), outra representação da solução (*s',m',l'*) deve ser avaliada. Esta compatibilidade é estimada pela Equação 5 que emprega a Distância de Manhattan [34] para estimar a proximidade da solução aos valores estipulados no SLA. Há quatro objetivos com escala de valores, portanto um sistema padronizado e necessário baseado em valores de *Gap*.

$$f(P[i]) = \left|\frac{C^c - C^*}{C^*}\right| + \left|\frac{T^c - T^*}{T^*}\right| + \left|\frac{A^c - A^*}{A^*}\right| + \left|\frac{C/h^c - C/h^*}{C/h^*}\right| \quad (5)$$

O espaço de busca e definido a partir do número de máquinas ajustadas pelo cliente, sendo uma quantidade 50% acima ou abaixo do número desejado de VMs são avaliadas. Por exemplo, se o cliente contratou 50 VMs, o intervalo possível para explorar permanece entre 25 e 75 VMs. Neste caso, a representação de soluções como (*s,m,l*)=(1,20,4) ou (*s,m,l*)=(0,0,75) é permitida desde $25 \leq (s + m + l) \leq 75$. Assim, todas as combinações possíveis de VMs dentro dessa faixa de 50% são avaliadas com o objetivo de obter o melhor valor para a Equação 5. O Algoritmo 1 (Fig. 1) descreve o algoritmo determinístico, no qual todas as combinações possíveis para (*s,m,l*) são avaliadas exaustivamente e o melhor é retornado.

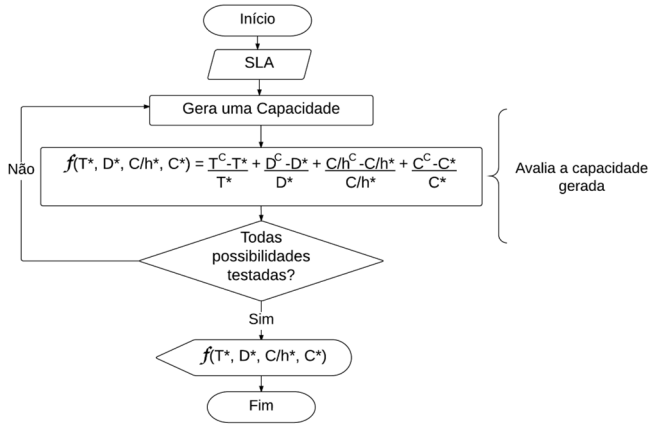

Figura 1.Representação do Algoritmo Determinístico.

O Algoritmo 2 (Fig. 2) mostra o μGA proposto. Este método é uma versão de algoritmo genético (AG) que opera com uma população de menor porte e emprega um critério de convergência que permite a reinicialização.

A população foi ajustada até que se estabeleceu em apenas 5 indivíduos, cada um dos quais representa uma possível configuração VM (*s,m,l*) para o cliente. A inicialização gera aleatoriamente os indivíduos (*s,m,l*) tais como; $min \leq (s + m + l) \leq max$, onde o intervalo possível é definido por [*min,max*].

Um torneio e aplicado para selecionar entre dois indivíduos um para participar do cruzamento. O BlX-α *crossover* [35] e aplicado para gerar novos indivíduos. No entanto, se $(s + m + l) \leq min$ ou $max \leq (s + m + l)$ para o novo indivíduo, a quantidade necessária é adicionada ou deduzida da última posição (*l*). Se não for suficiente, a quantidade também pode ser reduzida da posição *m*.

Não é usual ter operador de mutação em μGA, mas os testes de computação sobre este problema nos levam a incluí-lo com uma taxa de 20%.

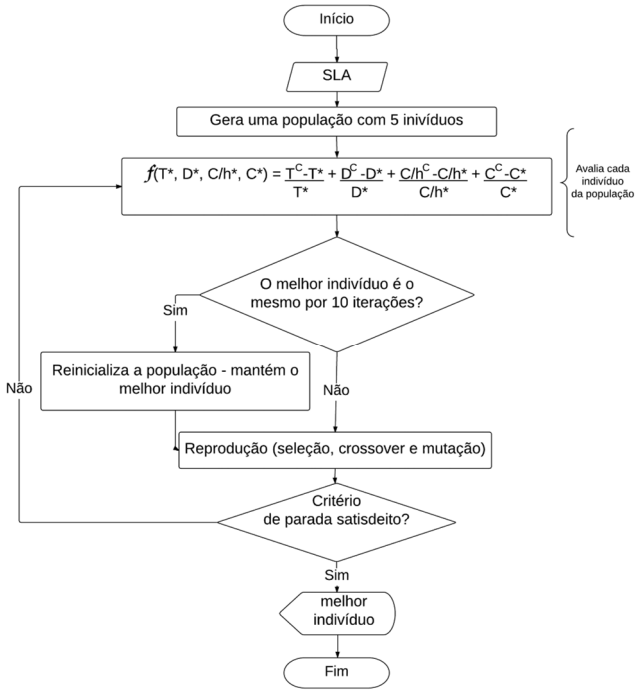

Figura 2.Representação do Algoritmo μGA.

O operador de mutação reinicializa completamente todos os valores de um indivíduo. O critério de convergência adotado foi baseado no pressuposto de que o melhor indivíduo não foi atualizado por mais de 10 iterações. Quando o critério de convergência é satisfeito, a população é reinicializada novamente e apenas o melhor indivíduo é mantido. O limite de tempo e o critério de parada.

V. RESULTADOS

Para execução dos experimentos foram utilizadas as configurações das VMs supracitadas na Seção IV, divididas homogeneamente de acordo com o cenário. Quanto a locação das VMs, elas foram divididas em *hosts* com configurações homogêneas, de forma que coubesse uma VM de cada tipo por máquina física. Os clientes foram organizados em forma de fila e atendido um por vez, a medida que chegava um novo cliente para contratar um SLA, ele era alocado no final da fila. Cada cliente possui uma carga de trabalho a executar, tais cargas eram di

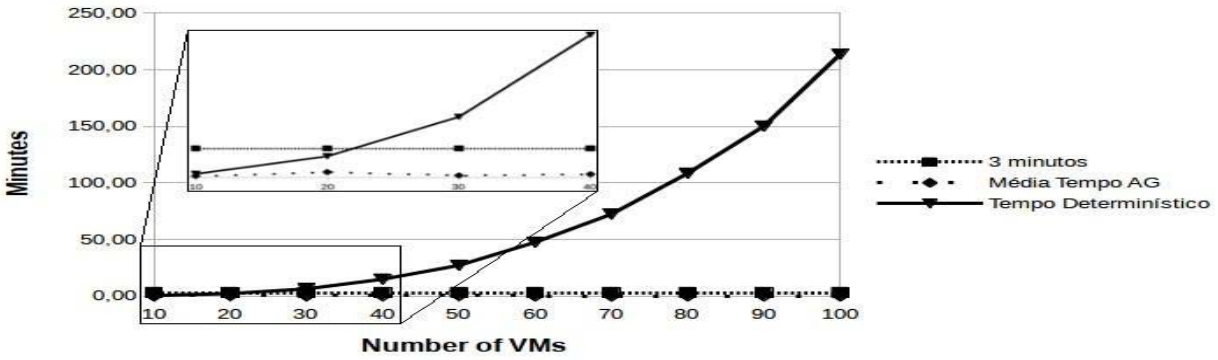

Figura 3. Tempos de respostas do modelo determinístico e do µGA resultantes do cenário de cenário aplicado de 10 até 100 VMs.

(entre 120000 e 130000 MIPS) e pesadas (de´ 160000 a 170000 MIPS), cargas do tipo I/O *bound*.

Os testes computacionais foram divididos em 10 cenários compostos por número diferente de VMs disponíveis em um *data center* conforme exibe a Tabela II.

Tabela II
EMPO DE RESPOSTA E NÚMERO DE COMBINAÇÕES OBTIDAS NA EXECUÇÃO DE 10 CENÁRIOS COM 10 ATÉ 100 VMS

Número de VMs	[*min,max*]	Tempo de Execução (min)	Número de Combinações
10	[2,8]	0.35	63
20	[5,15]	2.19	342
30	[7,22]	6.28	1330
40	[10,30]	14.83	2743
50	[12,37]	27.08	4912
60	[15,45]	47.53	9260
70	[17,52]	72.38	13823
80	[20,60]	108.56	19682
90	[22,67]	150.25	29790
100	[25,75]	213.86	39303

Relata-se o intervalo para explorar as soluções [*min,max*], o Tempo de Execução gasto e o número total de combinações avaliadas pelo algoritmo determinístico para cada cenário.

Para um *data center* com 10 VMs, assumiu-se que o cliente solicitou 5 VMS, então o intervalo a ser explorado é [2,8] neste caso. Pode-se observar que para 20 VMs, um tempo de execução bastante alto já é obtido em uma escala de minutos (2,19 minutos). O tempo aumenta exponencialmente para os outros cenários devido ao grande número de VMs. A execução demora mais de 20 minutos para 50 VMs.

O algoritmo determinístico e capaz de encontrar a solução ideal para este problema uma vez que explora completamente o espaço de solução. No entanto, estes resultados indicam que o método não é viável para propostas práticas, mesmo para um pequeno número de VMs.

O μGA foi executado 100 vezes para cada, considerando o máximo de 3 minutos como critério de parada. Esse tempo é razoável para responder a uma solicitação do cliente. Após as execuções foi realizada uma comparação da melhor solução do μGA com a solução ótima do algoritmo determinístico a fim de estabelecer uma taxa de sucesso alcançada pelo μGA.

solução ótima. O algoritmo μGA proposto foi capaz de encontrar as soluções ideais para todos os cenários em todas as 1000 execuções.

A Tabela III exibe o tempo médio com o desvio padrão gasto para encontrar a solução ideal. Essa taxa de correção foi obtida em menos de 1 minuto para todos os cenários para o μGA, em contraste com o método determinístico que levou mais de 1 minuto em todos os casos, exceto por VMs = 10.

TABELA III
TEMPO MÉDIO PARA ENCONTRAR SOLUÇÕES ÓTIMAS

VMs	Tempo em minutos
10	0.13 ± 0.15
20	0.54 ± 0.32
30	0.19 ± 0.19
40	0.32 ± 0.21
50	0.56 ± 0.24
60	0.17 ± 0.19
70	0.14 ± 0.17
80	0.17 ± 0.20
90	0.19 ± 0.21
100	0.16 ± 0.16

A Fig. 3 compara o tempo para todos os cenários. A linha contínua com o marcador triangular formando uma meia parábola representa o tempo obtido pelo algoritmo determinístico, a linha pontilhada com o marcador retangular representa o tempo limite de três minutos, o tempo considerado aceito pelo cliente. A linha pontilhada com o marcador na forma de um losango representa o tempo médio relatado para o μGA encontrar a solução ideal.

É possível observar pela Figura 3 que o algoritmo

determinístico obteve um tempo acima do que é tolerado pelo cliente, chegando a um tempo de complexidade exponencial. Em contraste, o μGA obteve uma solução dentro de um tempo consideravelmente menor do que o esperado pelo usuário, apesar do tempo limite ser alto (3 minutos), o μGA obteve a solução ótima em menos de um minuto para todas as soluções em menos de 30 segundos para a maioria das soluções.

VI. CONCLUSÃO

Este artigo abordou os desafios de fornecer ao cliente uma infraestrutura auto gerenciável com a provisão de QoS considerando o SLA acordado entre o cliente e provedor. A presente proposta mapeou alguns atributos de QoS que determinam os critérios do SLA. Também foram desenvolvidos e analisados algoritmos que fornecem uma reconfiguração otimizada da infraestrutura baseada em tais critérios. Os resultados evidenciam que o algoritmo µGA é eficiente e aplicável na solução do problema.

Em trabalhos futuros novos testes com um número maior de VMs. Serão efetuados bem como novos algoritmos metaheurísticos para que seja feita uma comparação entre eles. Também serão acrescentados outros atributos de QoS ao SLA e novas restrições ao problema como, por exemplo, um limite máximo de custo que o cliente consegue pagar.

AGRADECIMENTOS

Os autores agradecem CAPES, CNPq e FAPESP (processos número: 11/12670-5, 15/11623-4 e 16/14219-2) pelo apoio financeiro concedido para a elaboração deste trabalho.

REFERÊNCIAS

[1] P. Mell and T. Grance, “The nist definition of cloud computing,” 2012, disponível em: http://csrc.nist.gov/publications/nistpubs/800145/SP800-145.pdf. Último acesso em: 01/04/2017.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, no. 6, pp. 599 – 616, 2009.

[3] R. Buyya, J. Broberg, and A. M. Goscinski, Cloud Computing Principles and Paradigms.Wiley Publishing, 2011.

[4] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, “Cloud computing - the business perspective,” Decision Support Systems, vol. 51, no. 1, pp. 176 – 189, 2011.

[5] Intel-Corporation, “Cloud computing taxonomy and ecosystem analysis,” 2010, disponível em: http://www.intel.com/content/dam/doc/casestudy/intel-it-cloud-computing-taxonomy-ecosystem-analysis-study.pdf. Último acesso em: 01/04/2017.´

[6] S. Patidar, D. Rane, and P. Jain, “A survey paper on cloud computing,” in Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on, 2012, pp. 394–398.

[7] A. Kumbhare, Y. Simmhan, and V. K. Prasanna, “Exploiting application dynamism and cloud elasticity for continuous dataflows,” in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. ACM, 2013, p. 57.

[8] G. J. Woeginger, “Exact algorithms for np-hard problems: A survey,” in Combinatorial Optimization—Eureka, You Shrink! Springer, 2003, pp. 185–207.

[9] A. G. Kumbhare, Y. Simmhan, M. Frincu, and V. K. Prasanna, “Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure,” Cloud Computing, IEEE Transactions on, vol. 3, no. 2, pp. 105–118, 2015.

[10] A. E. Eiben and J. E. Smith, Introduction to evolutionary computing. Springer, 2003, vol. 53.

[11] L. V. Toscani and P. A. Veloso, “Complexidade de algoritmos,” ARTMED Editora SA, Porto Alegre–RS, 2008.

[12] B. G. Batista, J. C. Estrella, C. H. G. Ferreira, D. M. Leite Filho, L. H. V. Nakamura, S. Reiff-Marganiec, M. J. Santana, and R. H. C. Santana, “Performance evaluation of resource management in cloud computing environments,” PloS one, vol. 10, no. 11, p. 21, 2015.

[13] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, “Elasticity in cloud computing: a survey,” annals of telecommunications-annales des telecommunications, pp. 1–21, 2015.

[14] Amazon, “Scalingbasedonmetrics,” 2015, disponível em: http://docs.aws.amazon.com/autoscaling/latest/userguide/policy creating.html. Ultimo acesso em: 01/20/2017.

[15] ——, “Scaling the size of your auto scaling group,” 2015, disponível: http://docs.aws.amazon.com/autoscaling/latest/userguide/scaling plan.html. Ultimo acesso em: 01/20/2017.

[16] ——, “Dynamic scaling,” 2015, disponível em: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/asscale-based-on-demand.html. Ultimo acesso em: 01/20/2017.

[17] G. Soni and M. Kalra, “A novel approach for load balancing in cloud data center,” in Advance Computing Conference (IACC), 2014 IEEE International. IEEE, 2014, pp. 807–812.

[18] G. Kousiouris, T. Cucinotta, and T. Varvarigou, “The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks,” Journal of Systems and Software, vol. 84, no. 8, pp. 1270– 1291, 2011.

[19] J. Zhao, W. Zeng, M. Liu, and G. Li, “Multi-objective optimization model of virtual resources scheduling under cloud computing and it’s solution,” in Cloud and Service Computing (CSC), 2011 International Conference on. IEEE, 2011, pp. 185–190.

[20] L. Wu, S. K. Garg, and R. Buyya, “Sla-based admission control for a software-as-a-service provider in cloud computing environments,” Journal of Computer and System Sciences, vol. 78, no. 5, pp. 1280– 1299, 2012.

[21] M. Mohamed, D. Bela¨ıd, and S. Tata, “Extending {OCCI} for autonomic management in the cloud,” Journal of Systems and Software, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121216000030

[22] R. Mehrotra, S. Srivastava, I. Banicescu, and S. Abdelwahed, “Towards an autonomic performance management approach for a cloud broker environment using a decomposition–coordination based methodology,” Future Generation Computer Systems, vol. 54, pp. 195 – 205, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15000825

[23] M. Masdari, S. S. Nabavi, and V. Ahmadi, “An overview of virtual machine placement schemes in cloud computing,” Journal of Network and Computer Applications, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000291

[24] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Future Generation Computer Systems, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000689

[25] D. B. L.D. and P. V. Krishna, “Honey bee behavior inspired load balancing of tasks in cloud computing environments,” Applied Soft Computing, vol. 13, no. 5, pp. 2292 – 2303, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613000446

[26] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, “Cost optimized provisioning of elastic resources for application workflows,” Future Generation Computer Systems, vol. 27, no. 8, pp. 1011 – 1026, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000744

[27] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang, “Costefficient task scheduling for executing large programs in the cloud,” Parallel Computing, vol. 39, no. 4, pp. 177–188, 2013.

[28] B. Jennings and R. Stadler, “Resource management in clouds:

Survey and research challenges," Journal of Network and Systems Management, pp. 1–53, 2014.
[29] M. Guzek, P. Bouvry, and E.-G. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing," Computational Intelligence Magazine, IEEE, vol. 10, no. 2, pp. 53–67, 2015.
[30] S. Mustafa, B. Nazir, A. Hayat, S. A. Madani et al., "Resource management in cloud computing: Taxonomy, prospects, and challenges," Computers & Electrical Engineering, 2015.
[31] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and qos in cloud computing environments," in Parallel Processing (ICPP), 2011 International Conference on. IEEE, 2011, pp. 295–304.
[32] T. T. Huu and J. Montagnat, "Virtual resources allocation for workflowbased applications distribution on a cloud infrastructure," in Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. IEEE, 2010, pp. 612–617.
[33] J. M. Ko, C. O. Kim, and I.-H. Kwon, "Quality-of-service oriented web service composition algorithm and planning architecture," Journal of Systems and Software, vol. 81, no. 11, pp. 2079–2090, 2008.
[34] P. E. Black, "Manhattan distance," Dictionary of Algorithms and Data Structures, vol. 18, p. 2012, 2006.
[35] A. Eiben and J. Smith, Introduction to Evolutionary Computing, ser. Natural Computing Series. Springer Berlin Heidelberg, 2007. [Online]. Available: https://books.google.com.br/books?id=7IOE5VIpFpwC.

Luis Hideo Vasconcelos Nakamura e atualmente professor do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva. Recebeu seu diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas em 2006 pela Faculdade de Tecnologia de Taquaritinga (FATEC-TQ). Recebeu os títulos de mestre e doutor em Ciências Matemáticas e de Computação pelo Instituto de Ciências Matemáticas e de Computação (ICMC) pela Universidade de São Paulo (USP) em 2012 e 2017 respectivamente. Seus principais interesses estão nas áreas de Sistemas Distribuídos, Avaliação de Desempenho, Qualidade de Serviço, Web Semântica e Computação em Nuvem.

Leonildo Jose de Melo de Azevedo é atualmente estagiário de pesquisa na universidade de Leicester é estudante candidato ao título de Mestre em Ciências Matemáticas e de Computação pelo Instituto de Ciências Matemáticas e de Computação (ICMC) pela Universidade de São Paulo (USP). Leonildo é graduado em Ciência da Computação pela Universidade Estadual do Centro-Oeste (UNICENTRO) e suas áreas de interesse na pesquisa envolvem Computação Distribuída, Computação em Nuvem, Otimização, Metaheurísticas e Algoritmos Evolutivos.

Bruno Guazzelli Batista e professor do magistério superior da Universidade Federal de Itajuba. Doutor e mestre pelo Instituto de Ciências Matemáticas e de Computação (ICMC) na Universidade de São Paulo (USP), campus São Carlos. Graduado em Ciência da Computação pela Pontifícia Universidade Católica de Minas Gerais (PUC), campus Poços de Caldas. Realizou doutorado sanduíche com duração de um ano na Universidade de Leicester, Inglaterra. Experiência nas áreas de Sistemas Distribuídos, Tecnologia da Informação, Avaliação de Desempenho, TV Digital, Segurança de Sistemas de Informação e Qualidade de Serviço.

Rodolfo Ipolito Meneguette e professor do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva. Recebeu seu diploma de Bacharel em Ciência da Computação pela Universidade Paulista, Brasil, em 2006. Ele recebeu seu mestrado em 2009. Recebeu seu título de doutor pela Universidade Estadual de Campinas (UNICAMP), Brasil, em 2013. Atualmente, está realizando seu pós-doutoramento na Universidade de Ottawa. Os seus principais interesses são nas áreas de redes veicular e gestão da mobilidade de fluxo.

Claudio Fabiano Motta Toledo e graduado em Matemática Aplicada e Computacional pela Universidade Estadual de Campinas (1995), Mestre em Engenharia Elétrica pela Universidade Estadual de Campinas (1999) e doutor em Engenharia Elétrica pela Universidade Estadual de Campinas (2005). Atualmente é professor associado da Universidade de São Paulo (USP). Toledo possui experiência na área de Ciência da Computação, com ênfase em Sistemas Evolucionários, atuando nos seguintes temas: programação de produção, otimização, dimensionamento de lotes, metaheurísticas e algoritmos evolutivos.

Júlio Cezar Estrella recebeu o título de bacharelado em Ciências de Computação na Universidade Estadual de São Paulo - Júlio de Mesquita Filho (UNESP) em 2002, de Mestre e Doutor em Ciências de Computação pela Universidade de São Paulo (USP) em 2006 e 2010, respectivamente. Tem experiência em Ciência da Computação com ênfase em Arquitetura de Sistemas Computacionais, atuando nos seguintes temas: Internet de Coisas (IoT), Arquiteturas Orientadas a Serviços (SOA), Serviços Web, Avaliação de Desempenho, Sistemas Distribuídos, Redes de Computadores e Segurança de Computadores. Desde 2016, Júlio e professor associado do Instituto de Matemática e Ciência da Computação (ICMC - USP).

Stephan Reiff Marganiec é professor associado em Informática na Universidade de Leicester (Inglaterra). Trabalhou na indústria de informática na Alemanha e em Luxemburgo ocupando cargos de pesquisa na Universidade de Glasgow (ao mesmo tempo que cursava o doutorado/PhD) e na Universidade de Stirling. Ele presidiu a 8ª e 10ª *International Conference on Feature Interactions in Telecommunications and Software Systems* e foi PC *chair* na IEEE ICWS em 2016. Stephan lidera trabalhos na União Européia (UE) em projetos financiados pela Leg2Net, Sensoria e inContext focando a adaptação automática de serviços, seleção de servic¸os conscientes do contexto, composição de serviços baseada em fluxos de trabalhos e regras. Stephan é co-editor do *Handbook of Research on Service-Oriented Systems and Non-Functional Properties* e publicou mais de 80 artigos em conferências e revistas internacionais, além de ter atuado em um grande número de comitês de programas. Stephan foi nomeado professor convidado na Universidade de Petroleo da China e professor visitante em Lamsade na Universidade de Dauphine, em Paris. Ele foi eleito *Fellow of the BCS* (FBCS) em 2009 e é membro da ACM e da IEEE.

Regina Helena Carlucci Santana formada em Engenharia Elétrica pela Faculdade de Engenharia de São Carlos (1980), Mestrado em Ciência da Computação pelo Instituto de Ciências Matemáticas de São Carlos (1985) e Doutora em Eletrônica e Computação pela Universidade de Southampton (1989). Atualmente é professora associada da Universidade de São Paulo. Ela tem especialização em Ciência da Computação, com ênfase na Avaliação de Desempenho, atuando nos seguintes tópicos: medição de desempenho, simulação, simulação distribuída, tarefas e programação de processos e computação paralela. Outro tópico de interesse na pesquisa é a Arquitetura de Sistemas Computacionais Distribuídos envolvendo Cluster, Grid, Cloud Computing e outros.

Marcos José Santana graduado em Engenharia Elétrica Eletrônica pela Faculdade de Engenharia de São Carlos (1980), Mestrado em Ciência da Computação pelo Instituto de Ciências Matemáticas de São Carlos (1985) e Doutorado em Eletrônica e Computação pela Universidade de Southampton (1989). Atualmente é Professor Associado da Universidade de São Paulo. Ele tem experiência em Ciência da Computação, com ênfase na avaliação de desempenho, atuando nos seguintes tópicos: avaliação de desempenho, serviços web, computação em *cluster*, computação em grade, computação em nuvem, agendamento de processos, computação paralela, simulação e balanceamento de carga para sistemas distribuídos. Coordenador de Engenharia de Computação do ICMC desde 2002 até 2011 e Chefe do Departamento de Sistemas de Computadores desde 2010.