Algoritmos de Otimização na Alocação de Recursos em Datacenters Prestadores de Serviços de Nuvem*

Erickson Leon Kovalski†

Resumo: O uso de recursos na nuvem ganhou enorme popularidade na indústria, principalmente por sua elasticidade, permitindo ao cliente adquirir serviços automaticamente e sob demanda, associado ao modelo de precificação proporcional ao uso. Porém, o desempenho dos recursos na nuvem é ainda um fator limitante para o crescimento deste modelo de negócio, podendo levar o cliente a procurar outro provedor ou criar seu próprio datacenter local por incerteza quanto à capacidade de o provedor cumprir o Acordo de Nível de Serviço (SLA). Neste trabalho, procedemos uma revisão de literatura sobre otimização em datacenters. Sugerimos uma classificação para os algoritmos de otimização. Listamos as técnicas recentemente desenvolvidas, apresentando um panorama sinótico do *estado-da-arte* em otimização de datacenters a fim de superar desafios para oferta de serviços na nuvem.

Palavras-chave: Datacenter. Algoritmo. Heurística.

1. INTRODUÇÃO

O rápido crescimento do modelo de computação na nuvem, que vem ocorrendo há mais de uma década, trouxe consigo o desafio de acomodar clientes com diferentes demandas dentro de uma mesma estrutura física. Para o modelo ser economicamente viável, do ponto de vista do cliente, a nuvem deve se apresentar como uma fonte ilimitada e uniforme de recursos computacionais (*thin provisioning*), seguro, de baixa latência e alta disponibilidade. Desta forma, o cliente pode contratar à medida que necessite dos recursos. Ao mesmo tempo, para um Prestador de Serviço de Nuvem – *Cloud Service Provider* (CSP), os equipamentos consomem energia elétrica, demandam controle de temperatura e umidade, geram impactos no meio-ambiente, sofrem manutenção e são substituídos, ademais, diferentes aplicações de usuários fazem uso irregular do tráfego de rede ou do tempo de processamento, entre outros. Em vista disto, um certo ponto de equilíbrio precisa ser encontrado o qual minimize as faltas quanto ao Acordo de Nível de Serviço – *Service Level Agreement* (SLA) ao mesmo tempo que maximize a rentabilidade de um *site* ou de um conjunto de *sites* para o CSP.

Assim, a gestão dos recursos do datacenter ocupa uma posição central para a redução dos custos operacionais, manutenção da clientela e, consequente, retorno do investimento. E, para tanto, o uso de virtualização e agendamento de tarefas é entendido como a forma

---

* Artigo apresentado como Trabalho de Conclusão do Curso de Especialização em Datacenters: Projeto, Operação e Serviços, da Universidade do Sul de Santa Catarina - UNISUL, como requisito parcial para a obtenção do título de Especialista. Orientador Prof. Ms. Luiz Otávio Botelho Lento

† Aluno do curso de Especialização Datacenters: projeto, operação e serviços pela Universidade do Sul de Santa Catarina.

mais eficiente de aproveitar os equipamentos de hardware disponíveis. Porém, efetivamente estimar a demanda de recursos virtualizados tem sido um desafio, muitas vezes levando a um resultado sub-ótimo e insatisfatório tanto para o cliente quanto ao prestador.

O consumo de energia de uma máquina virtual – *Virtual Machines* (VM), por exemplo, é uma métrica que permite melhorar a distribuição de recursos computacionais. No entanto, nota-se que o consumo estimado pela VM não se traduz no consumo do processo subjacente na máquina física, devido à heterogeneidade dos equipamentos na camada de *hardware;* ademais, o uso de soluções de monitoramento por *software* e medidores de energia físicos possuem correspondência errática. Novas abordagens são exigidas, capazes de atender à solicitação dinâmica de recursos típica de um datacenter que fornece serviços de processamento e armazenamento na nuvem. Pois, ao se considerar o grau de complexidade que tais ajustes podem implicar, fica evidente que soluções tradicionais não conseguem captar as interrelações necessárias para obter o resultado desejado. O número de variáveis envolvidas torna difícil a abordagem do problema dentro de um tempo adequado.

Desafios formalmente similares, porém, são encontrados por organismos biológicos sociais assim como aos frequentemente abordados pela matemática e estatística. Sabe-se, de antemão, que a solução ótima não é sempre possível de ser alcançada, e que restrições de tempo computacional e de memória operacional podem rapidamente se tornar proibitivas. Porém, dados os *constraints* inerente ao problema, é possível muitas vezes se encontrar soluções suficientemente próximas da ideal utilizando um algoritmo apropriado.

Um exemplo é inspirado na observação do comportamento de abelhas melíferas na busca de pólen ou néctar. Algumas abelhas da colônia passam a explorar as redondezas buscando pelo alimento, avaliando a produtividade de cada fonte encontrada. As abelhas que encontram boas fontes de alimentos vão para uma região da colmeia onde realizam a dança, comunicando a abelhas então ociosas a localização do recurso, que se unem para exploração deste. À medida que mais abelhas retornam à colmeia, mais abelhas serão recrutadas através da dança para explorar o recurso recém descoberto, porém, à medida que o recurso escassear, as abelhas reduzirão o grau de abertura do movimento na dança, sinalizando que a estimativa de produtividade foi reduzida. Neste contexto, outras abelhas que retornem de fontes mais promissoras receberão mais atenção, recrutando novas operárias para explorar o recurso.

Este modelo foi formalizado por um algoritmo simplificado, o qual é implementado por um número determinado de agentes, interagindo em um espaço virtual. Pode-se, por analogia, entender recursos ociosos de um datacenter como flores promissoras, para os quais serão migradas VM até o recurso deixar de ser interessante para os agentes. À medida que novos recursos se tornam ociosos, estes passam a ser mais interessantes para hospedar um novo recurso virtualizado, e isto ocorre dinamicamente, conforme a oferta e a demanda.

Um algoritmo similar pode ser aplicado à otimização do roteamento de telecomunicações, como o baseado no comportamento de formigas. Formigas deixam um feromônio ao se deslocarem, este é utilizado para retornarem ao formigueiro, mas também para outras formigas refazerem os passos na outra direção, rumo à fonte de alimento. À medida que o feromônio evapora, tal trajeto passa a ser menos interessante para outras formigas, reduzindo a afluência por esta via. Este modelo é particularmente útil para se encontrar a menor distância prática para um trajeto entre dois pontos, quando muitos caminhos são possíveis. Isto é possível pois o caminho mais curto será percorrido mais vezes por formigas (ou agentes), quando comparado a um caminho mais longo, concentrando mais feromônio e, probabilisticamente, inclinando futuras formigas a percorrerem o trajeto mais frequentado. Este algoritmo pode ser adaptado para encontrar dinamicamente o trajeto mais eficiente entre os pontos de uma rede de comunicações, o com menor latência, ou o com maior largura de banda disponível.

Problemas de otimização emergem frequentemente em um datacenter. Da qualidade das soluções depende a lucratividade e a competitividade do datacenter. Este é um tema muito pesquisado e muitos artigos científicos dedicam-se a explorar diversas técnicas para se aprimorar soluções já existentes. Porém, tais técnicas estão longe de ser triviais, usualmente requerendo um grande poder computacional que é limitante quanto à qualidade da solução. Isto é, soluções ótimas podem levar um tempo proibitivamente longo para serem computadas, ou requerer recursos de memória que excedem a disponível para a operação do datacenter. Portanto, é essencial que se desenvolvam técnicas capazes de convergir em direção a resultados suficientemente próximos do ótimo rapidamente e que façam uso de menor quantidade de recursos computacionais.

Buscamos neste trabalho listar as técnicas recentemente desenvolvidas, através de revisão de literatura, de forma a construir um panorama sinóptico do estado-da-arte em otimização de datacenters. Para tanto propomos uma classificação das técnicas por similaridade de abordagem e focamos nos principais desafios apresentados pela revisão de literatura para oferta de serviços na nuvem.

1.1 Motivação e trabalhos relacionados

Dentre os problemas mais frequentes de um datacenter, duas grandes categorias emergem, uma que tem como objetivo otimizar a performance e outra que pretende otimizar o uso de energia. Estes objetivos antagonizam um ao outro, na medida em que maior performance é requerida, custos de energia para alimentar os equipamentos e proporcionar a necessária ambientação crescem de forma não linear. Em contrapartida, à medida que ações são postas em prática para reduzir os custos energéticos, performance é sacrificada concentrando tarefas em servidores ativos ou fazendo uso de limitação dinâmica de frequência e tensão – *Dynamic Voltage and Frequency Scaling* (DVFS) nos equipamentos.

O mercado de datacenters é inerentemente competitivo, e isto é intensificado pelo modelo de nuvem: internamente, pela facilidade de contratar os serviços sob demanda e migrar máquinas virtuais; e externamente, devido à atual perspectiva no desenvolvimento de produtos de software como algo em teste permanente (*perpetual beta*), onde numerosas

iniciativas são postas à prova no mercado e sua continuidade, abrangência e aprimoramentos são reavaliados constantemente. Desta maneira, possibilidades de atuação, tanto para o CSP quanto para seus clientes atuais e prospectivos, vêm se ramificando e ampliando. O datacenter, portanto, tem se tornado cada vez mais complexo, tanto pela quantidade de equipamentos que abriga e pelo volume de dados que armazena e trafega, quanto na variedade de funções que passa a assumir.

Ajustar o datacenter para o seu melhor rendimento implica ajustar as variáveis, as quais podem ser descritas como Qualidade de Serviço – *Quality of Service* (QoS), metas ambientais de conservação e energia, aproveitamento dos recursos físicos, entre outros. Além disto, é preciso levar em consideração a quantidade de hardware a ser gerenciado, desde servidores a equipamento de rede, cada qual com uma variedade de parâmetros a serem ajustados individualmente. Isto, se apresenta como um desafio para o qual o atual *know-how*, na forma de políticas e boas práticas, assim como as soluções prontas, *off-the-shelf* são insuficientes. Com efeito, o melhor aproveitamento de máquinas físicas – *Physical Machines* (PM), o melhor uso dos recursos de rede, a melhor distribuição de tarefas, são objetivos que ocultam uma admirável intratabilidade. Isto é, tais objetivos dependem da solução de um problema subjacente que se torna progressivamente mais custoso à medida que as dimensões consideradas crescem. Assim, adentramos o campo da complexidade computacional, em cujo centro está, em aberto, um dos *Problemas do Prêmio Millennium*, P vs. NP, que pergunta se a todos os problemas os quais um algoritmo pode *verificar* rapidamente a solução, existe um também um algoritmo que pode *encontrar* a rapidamente solução.

Virtualização é uma ferramenta que facilita o gerenciamento de um datacenter e pode auxiliar na redução do *downtime* e do consumo de energia, aumentar a segurança e facilitar o *back up* e a transferência de serviços entre equipamentos. Há principalmente dois recursos que são virtualizados: servidores e redes. Ainda assim, o conceito pode se estender desde clusters a datacenters inteiros, ou a redes que englobam um conjunto de datacenters distribuídos geograficamente. Muitos trabalhos têm se dedicado a consolidação de VMs em um reduzido número de PMs. Idealmente se deseja concentrar um número de VMs no mesmo recurso físico subjacente de forma a otimizar o poder de processamento e consumo de energia. Quando o volume de trabalho aumenta ou reduz, VMs são realocadas dinamicamente (*Live Migration*), com um mínimo de downtime.

Neste sentido, emerge o problema de distribuir a carga de trabalho, ou *workload*, através dos servidores, clusters ou datacenters. Isto está vinculado ao adequado balanceamento de carga (*load balancing*), que procura evitar a sobrecarga da rede pelo melhor aproveitamento da largura de banda, e o eficiente agendamento de tarefas (*scheduling*), que por sua vez busca encontrar a adequada distribuição do processamento através do poder computacional disponível. Devido à importância que este problema impões aos CSPs, um grande número de trabalhos tem sido publicado no sentido de sistematizar as técnicas de otimização utilizadas.

Em (1) são classificadas conforme o método de otimização, que pode ser exato, heurístico ou meta-heurístico. a) Dentre os métodos de otimização são classificados como

exatos estão Programação Estocástica, Programação Linear, Programação Não-Linear, Programação com Restrições, Programação Quadrática e Teoria dos Jogos; b) Dentre os métodos heurísticos, *First Fit Decreasing* (FFD), *Best Fit Decreasing* (BFD), *Next Fit*, *Randon Fit*, *Least Full Fit*, *Most Full Fit*, Produto Interno, e *Minimizing Angle*; e c) dos métodos meta-heurísticos, Algoritmo Genético - *Genetic Algorithm* (GA), Grouping Genetic Algorithm (GGA), Recozimento Simulado – *Simulated Annealing* (SA), Método de Enxame de Partículas – *Particle Swarm Optimization* (PSO), *Tabu Search*, e otimização híbrida.

Já (2), focando no modelo IaaS (Infraestructure as a Service), procede a uma classificação com base na técnica de otimização, subdividindo em a) heurística de empacotamento (*bin packing*), como FFD, BFD; b) otimização baseada em biologia, como otimização da colônia de formigas – *Ant Colony Optimization* (ACO) e GA; c) programação linear, como *LP- Relaxation* e *Integer Linear Programing* (ILP); d) programação com restrições e e) SA.

Em (3) são classificados algoritmos em a) exatos, b) heurística para único datacenter, c) heurística para Multi-IaaS, e d) heurísticas para outras formulações, que incluem algoritmos para nuvens híbridas e multi-datacenter. Considera que algoritmos exatos não são capazes de execução em tempo polinomial ou pseudo-polinomial devido ao problema ter em seu centro o *bin packing*, reconhecido por ser NP-hard. Não obstante, lembra que para casos particulares pode ser possível adaptar algoritmos exatos.

O trabalho de (4) Categoriza os algoritmos de alocação de VM conforme o objetivo em baseados no consumo energético, que buscam realizar o mapeamento VM-PM para máxima eficiência energética ou baseados em QoS, ou seja, que procuram garantir o máximo dos requisitos de qualidade de serviço. E classifica a abordagem em a) Programação com Restrições, b) *Bin packing*, d) Programação Linear Estocástica e c) Algoritmos Genéticos, neste último caso incluindo ACO. Com base na diversidade de abordagens encontrada e na impossibilidade de um algoritmo sobressair como melhor que os outros, sugerem que se deve pesquisar algoritmos que busquem minimizar o comprometimento de performance em função da eficiência energética.

Em (5), um trabalho que busca otimizar o *workflow* tendo como objetivo reduzir os custos, aponta que o método de otimização tem o maior impacto no custo pois diretamente se vincula o mapeamento de recursos às tarefas. Tais métodos são classificados em a) Heurísticos, b) Meta-heurísticos, c) *Greedy*, d) *Clustering*, e) *Fuzzy* e f) Modelagem Matemática. Os autores notam que os métodos meta-heurísticos vem sendo usados, porém com comprometimento do tempo de execução.

A partir da classificação de (6), que tem como objetivo otimizar para eficiência energética através de alocação dinâmica de VMs, emergem quatro classes de algoritmos, a) Programação Estocástica Inteira, b) Algoritmo Genético c) *Bin Packing*, d) Programação com restrições. Nota que o melhor resultado em eficiência energética é obtido concentrando VMs em um mínimo de servidores e mantendo os demais totalmente ociosos.

2. Abordagem heurística do Datacenter

Há problemas para os quais encontrar uma solução ótima ou exata requer muitos passos, que podem ser entendidos como intensos computacionalmente ou intensos no uso de memória, mas ainda assim são essencialmente tratáveis. Um exemplo seria dizer se certo número é primo (7). Trata-se de um problema de decisão, isto é, a resposta é um "sim" ou "não". Para este caso específico existem algoritmos para os quais é garantido obter tal resposta em tempo polinomial, o que significa dizer que se precisarmos decidir sobre um número primo muito grande, basicamente bastaria um computador rápido para se obter o resultado. No entanto, estima-se que a maioria dos possíveis problemas não possui solução algorítmica, como o *Halting Problem*, explorado por Alan Turing em seu artigo seminal (8). Entre estes dois extremos há diversas classes de problemas que, ao mesmo tempo que não são passíveis de serem resolvidos em um tempo razoável, ainda possuem, em tese, solução algorítmica (9). Isto é, uma Máquina de Turing seria perfeitamente capaz de encontrar a solução, dado tempo e papel infinitos para realizar as operações necessárias (10).

Abordando como classe, externamente teríamos um plano infinito *O* onde estariam todos os problemas ou funções possíveis. Um problema em *O* pode ser resolvido por alguma Máquina Oráculo em até um passo (*Fig. 1*). Neste espaço encontraríamos uma classe de problemas *R*, os quais podem ser resolvidos por algoritmos recursivos por uma máquina de turing, em tempo e memória infinitos (11). Internamente temos os problemas na forma c^n, onde *c* é uma constante e *n* um polinômio, isto significa que o tempo de execução varia exponencialmente em função da entrada. Internamente a próxima classe é *NP*, estes podem, no entanto, ser resolvidos rapidamente por uma Máquina Não-Determinística, mas uma máquina de turing, necessita explorar todo o espaço de soluções sequencialmente, em grande parte dos casos demandando um tempo extraordinariamente longo (10). A classe internamente seguinte, *P*, limita-se a problemas que podem ser resolvidos em tempo polinomial, n^c, por uma Máquina de Turing. Esta é a classe dos problemas que são considerados tratáveis, no sentido de existir um algoritmo rápido para se chegar a solução (10,12). Porém apenas um pequeno conjunto, *P'*, contém os problemas que, de fato, são viáveis de serem resolvidos de um ponto de vista prático.

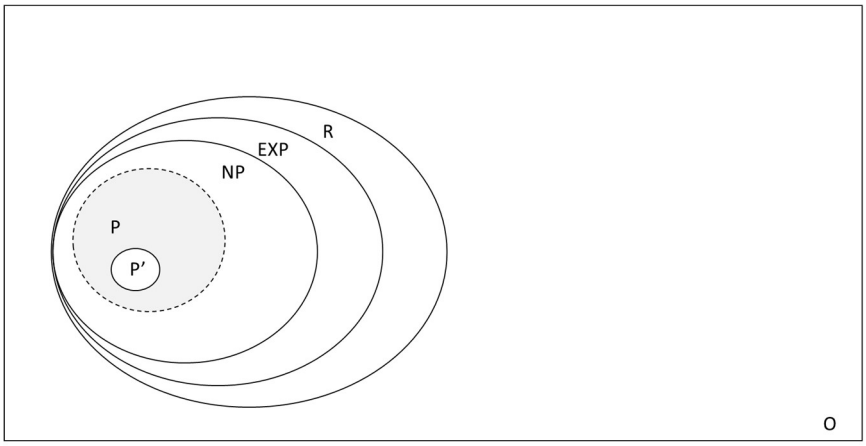

Figura 1: Classes de Problemas

Embora não se saiba se P=NP, é em geral presumido que este não é o caso. Um problema é dito NP-hard se for ao menos tão complexo quanto o mais complexo na classe NP. Isto implica, portanto, que um problema em EXP-hard é NP-hard, mesmo que não esteja em NP. Cabe notar que, igualmente, não podemos afirmar NP=EXP como verdadeiro ou falso (10).

Para se afirmar que um problema novo B é NP-hard, prova-se reduzindo um problema conhecido A NP-hard para ele, com isto significando que B é ao menos tão complexo quanto A. Se um problema A que é sabidamente hard na classe NP, pode ser reduzido para outro, problema B, da mesma classe, então B é dito NP-completo desde que esta redução se dê em tempo polinomial. Desta maneira, um problema NP-completo pode ser transformado em qualquer outro NP-completo e possuirá a mesma solução. Cabe lembrar que, de posse da solução a um problema NP-Hard, e que esteja ele mesmo em NP, verificar a correção do mesmo é possível em tempo polinomial (esta é, de fato, a definição de NP) (10).

Problemas de decisão, como comentado acima, possuem uma resposta do tipo “sim” ou “não”. Neste sentido, não há muito que se possa fazer quanto a melhorar a solução de um problema de decisão NP-hard, ou o resultado é obtido e se pode certificar sua validade, ou não é obtido e se continua a explorar o espaço de pesquisa. Por outro lado, problemas de otimização procuram o melhor resultado para um dado problema, isto é, produz-se resultados parciais, que podem estar longe de serem ótimos, e tempo adicional de processamento é utilizado com a finalidade de melhorá-los. Isto abre a possibilidade de se deixar de lado a procura pelo resultado ótimo, já que este se mostra inviável, e desenvolverem-se técnicas para procurar um resultado que seja *bom o suficiente*. Em outras palavras, quando métodos exatos se mostram infrutíferos pode se lançar mão de métodos heurísticos para se chegar a uma resposta aproximada.

Dada a complexidade que a operação de datacenters impõe, uma abordagem que permita o uso de algoritmos conhecidos para a solução dos problemas de otimização é bastante vantajosa. Apesar da impossibilidade técnica para se encontrar soluções exatas para os problemas NP-hard, em certas condições, tais problemas podem ser adequadamente abordados por algoritmos exatos. Também pode ser o caso de soluções aproximadas serem boas o suficiente para questões práticas. Neste sentido é importante classificar precisamente o que se pretende otimizar, descrevendo de modo que seja possível aplicar um método conhecido. Pelo visto antes, ao se deparar com um problema que sabidamente está em NP, provar a *hardness* do problema é fundamental na escolha da heurística a ser aplicada (13). Em (14), por exemplo, que busca alocar eficientemente VMs para um cluster, primeiro reduz o problema de teoria dos grafos k-clique, já provado NP-hard na literatura, à descrição apresentada. A partir desta prova, procede o ajuste de um algoritmo aproximado baseado em grafos para alocar VMs. Em outro exemplo, (15) invoca o problema do caixeiro viajante – *Traveler Salesman Problem* (TSP) – como modelo, ao procurar minimizar o tráfego de carga entre VMs. Igualmente, procede primeiro a uma prova de que se trata de um problema ao menos tão complexo quanto TSP, e que é possível reduzir sua abordagem a TSP em tempo polinomial (16). Desta

maneira, a aproximação dos desafios apresentados por um datacenter a problemas conhecidos pode ilustrar, modelar, assim como, sugerir soluções práticas.

O problema do caixeiro viajante (TSP) é apresentado como: dadas *n* cidades e suas distâncias entre si, encontrar o menor circuito que visita todas cidades exatamente uma vez. Embora esteja a relacionado a como encontrar o circuito hamiltoniano em um grafo (17), cuja teoria subjacente remonta a Hamilton e Kirkman (1856), e a descrição do problema seja encontrada em um manual para vendedores, datado de 1832, a formulação matemática deste vai apenas ser dada a partir do trabalho de Menger, em 1930. Em 1949, Julia Robinson, descrevendo iniciativas mal sucedidas de se resolver o problema em um relatório da RAND, pela primeira vez que se tem notícia se refere a ele como *Traveling Salesman Problem*. Merrill M. Flood, em 1956, comenta que o trabalho sobre o TSP revelou uma complexidade que irá requerer um uma abordagem diferente para ser tratado e, ao mesmo, tempo relaciona TSP ao problema de agendamento, apresentado por George Feeney em 1954, na Universidade de Columbia. Nota que uma solução bem-sucedida para um seria aplicável ao outro.

Nos anos 60s, nota (18), uma classe de problemas para os quais uma solução em tempo polinomial não era conhecida começou a se evidenciar. Estes eram na maior parte problemas de otimização cuja quantidade de soluções possíveis rapidamente crescia e não havia uma forma conhecida de se escolher, dentre estas, a melhor. São problemas que abordar por simples força bruta não era, portanto, viável e isto levou a se suspeitar que tal solução não existia de fato. Stephen Cook, em 1971, e Leonind Levin, em 1973, demonstraram que estes problemas fazem parte de uma classe que viria a se chamar NP-completos. Richard Karp (16,19) demonstrou que oito problemas fazem parte desta classe, TSP dentre eles, e que portanto não existe possibilidade de se resolver TSP em tempo polinomial – e assim não faz sentido procurar por esta solução, exceto se P=NP – ao mesmo tempo que uniu os diversos problemas desta classe através demonstração da redutibilidade de um a outro.

Embora à primeira vista as questões de complexidade algorítmica façam parte de uma área da matemática que se apresenta como um campo abstrato o qual parece desvinculado dos problemas reais do cotidiano, a aplicabilidade de soluções para estas questões no ambiente de datacenter indicia a intensa conexão entre os avanços na pesquisa teórica e as melhorias nas aplicações práticas.

2.1 Proposta de Classificação dos Algoritmos de Otimização

As técnicas de otimização em um datacenter serão divididas neste trabalho em tradicionais, heurísticas, meta-heurísticas, baseadas em *machine learning* (ML), baseadas em teoria dos jogos e estatísticas (*Fig. 2*). As técnicas que aqui denominamos tradicionais incluem o método do máximo declive (*gradient descent*), o método de Newton, a programação linear, a programação com restrições, a programação não linear, a programação inteira – *Integer Linear programming* (ILP), a programação inteira mista – *Mixed integer linear programming* (MILP), a Programação Quadrática e *Simplex*, mas

também as técnicas baseadas em políticas. As técnicas matemáticas são úteis em muitos casos particulares, porém costumam demandar muito tempo computacional, pois se trata de problemas NP-hard. As políticas não tratam de problemas de otimização, antes, são mais uma coleção de sugestões e boas práticas sistematizadas e não nos deteremos nestas.

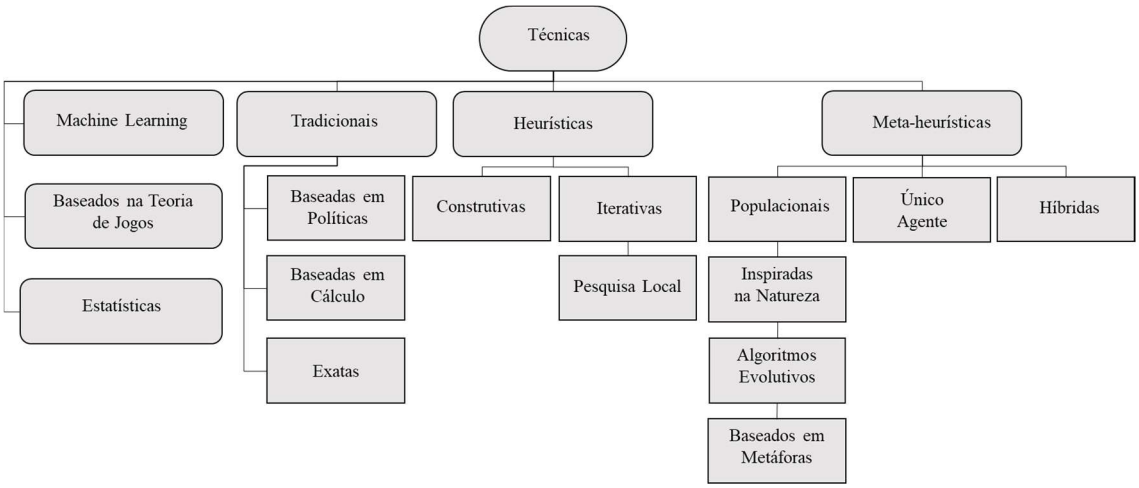

Figura 2: Proposta de Classificação de Técnicas de Otimização

As técnicas heurísticas (*Tab. 1*) pretendem dar um passo além, evitando explorar espaços de pesquisa nos quais há pouca chance de se encontrar o resultado ótimo. Entre elas podemos distinguir as construtivas, baseadas em variações de um algoritmo ganancioso (*greedy*), como *First Fit* (FF), ou iterativos, que se caracterizam pela exploração aleatória de possíveis soluções sub-ótimas. Através da comparação entre as possíveis soluções a heurística escolhe explorar as que se apresentam como mais promissoras (isto é, trata-se ainda de algoritmos *greedy*). O que pode levar o algoritmo a ficar preso em um máximo ou mínimo local. Exemplos são *Depth First Search* (DFS), *Breadth First Search* (BFS), *A**. Tais métodos compuseram o cânone da inteligência artificial simbolista até os anos 60.

Tabela 1: Técnicas Heurísticas (20)	
Construtivas	First Fit (FF) First in First out (FIFO) Last in Last out (LILO)
Iterativas	Breadth First Search (BFS) Deep First Search (DFS) A* Randon Search Dijkstra's Hill Climbing MiniMax

As técnicas meta-heurísticas pretendem aprimorar as heurísticas evitando que o espaço de procura da solução acabe restrito a um máximo/mínimo local. *Tabu Search* (TS) e SA são métodos que procuram a solução através de ajustes no espaço de pesquisa. O último (21), utiliza a analogia de redução da energia, ou resfriamento, para explorar o

espaço de pesquisa de forma que, dado um sistema que apresente vários máximos (ou mínimos) locais, estes tenham a chance de ser explorados, aumentando a possibilidade de se chegar a um resultado mais próximo do ótimo. Já a TS (22) aborda o problema de uma direção diferente, e o faz moderando a ganância na procura por soluções, permitindo que o algoritmo aceite soluções de menor qualidade na expectativa que isto possa conduzir a outros *extrema*. O termo tabu se refere à necessidade de marcar espaços já explorados como interditados para evitar que o algoritmo fique revisitando a mesma possibilidade. Isto não é necessário em um algoritmo *greedy* pois este vai sempre na direção do melhor resultado vizinho.

Técnicas meta-heurísticas populacionais exploram o espaço de pesquisa através de uma população de candidatas à melhor solução. Um exemplo é PSO (23), que utiliza a ideia de partículas navegando no espaço de pesquisa. Cada partícula procura a melhor posição-solução. Resultados parciais influenciam o conjunto, levando partículas com resultados inferiores a migrarem em direção daquelas próximas que apresentam melhores resultados. A aposta aqui é que a solução que atrai mais partículas é a melhor encontrada, porém não há garantia de que esta seja, de fato, a melhor possível.

Heurísticas baseadas na evolução procuram simular a evolução natural dentro do espaço de pesquisa no intuito de se obter a solução que possui maior *fitness*, isto é, a que representa o melhor resultado. Assim como se dá na natureza, este método facilmente converge a máximos/mínimos locais ignorando soluções possivelmente melhores. Para contornar tal limitação, as taxas de mutação e *crossover* são ajustadas de acordo com o problema para permitir transpor os vales do espaço de pesquisa (24).

Um tipo de meta-heurística que tem se diversificado recentemente é o baseado em metáforas (*Tab. 2*). Este consiste na observação de um comportamento biológico para a seguir modelá-lo, usualmente adaptando uma técnica conhecida, a fim de abordar problemas de otimização. Um famoso exemplo é ACO (25,26). Neste, entende-se o espaço de pesquisa como o que separa a colônia de formigas do alimento e indivíduos devem cooperar através da deposição de feromônios para encontrar o trajeto mais curto. Outras técnicas irão combinar, por exemplo, heurísticas já existentes com técnicas tradicionais ou ML; estas chamamos aqui híbridas, como exemplo temos os algoritmos meméticos.

Tabela 2: Exemplos de Meta-Heurísticas Metafóricas de Inspiração Biológica

Marcos Dorigo, 1992 (25)	Ant Colony Optimization (ACO)
Davis Karaboga, 2005(27)	Artificial Bee Colony (ABC)
D.T. Pham et ali, 2005 (28)	Bee Algorithm
M. M. Eusuff e K. Lansey, 2003 (29)	Shuffled Frog Leaping Algorithm (SFLA)
Geem, Kim, Loganathan, 2001(30)	Harmony Search
Atashpaz-Gargari e Lucas, 2007 (31)	Imperialist Competitive Algorithm
Xin-She Yang, 2010 (32)	Bat Algorithm
Xin-She Yang, 2008 (33)	Firefly Optimization
Li LX, Shao ZJ, Qian JX (2002) (34)	Artificial Fish Swarm Algorithm (AFSA)

Além destas técnicas, também são utilizados Métodos Estatísticos como Média Móvel, útil em geral para realização de predições; alguns métodos que utilizam Teoria dos Jogos, que desenvolvem-se em torno de estratégias antagônicas as quais visam atingir o *equilíbrio de Nash*; e ML, o qual inclui *Support Vector Machines* (SVM) e técnicas mais complexas de otimização que incluem uma fase de treinamento.

3. Metodologia

Realizamos uma revisão sistemática de literatura nas bases de dados convencionais de artigos acadêmicos em língua inglesa publicados nos últimos cinco anos em periódicos revisados por pares. A pesquisa foi delimitada pelas palavras-chave "data center" e "optimization", em todos os casos.

Utilizamos as bases disponibilizadas pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) as quais estavam operacionais através do portal de periódicos durante o período de elaboração desta pesquisa. Dentre estas foram selecionadas somente as que contém artigos relevantes à esta pesquisa e que atendem aos critérios mencionados acima. Desta forma, as bases selecionadas foram a *EBSCO*, a *MathSciNet*, a *Oxford Journals*, a *PROQUEST*, a *ScienceDirect*, a *Scopus*, a *SpringerLink* e a *WebOfScience*. A pesquisa com as palavras-chave e delimitações propostas resultou em N=1969 artigos individuais. Para a remoção de artigos duplicados utilizamos o software *JabRef* e passamos a verificar o DOI e a recolher os resumos e palavras-chave, assim como o *link* para cada artigo.

A seguir, os artigos cujo objeto não era datacenters ou que não mencionavam um tratamento heurístico ou algorítmico em seu resumo foram excluídos. Isto resultou em 1385 artigos.

Procedemos então à verificação da qualidade dos artigos, através da pontuação do periódico em que foram publicados segundo o *Scimago Journal & Country Rank* (SJR). Escolhemos apenas artigos de *journals* do primeiro quartil para a área de Ciência da Computação, Matemática ou Engenharias, isto é, apenas artigos publicados em 25% dos periódicos mais citados. Paralelamente utilizamos o índice JCR≥1 para casos em que houve ambiguidade ou dados recentes indisponíveis. Com isto o N de artigos para leitura de texto completo reduziu para 330.

Na fase seguinte a maior parte dos artigos foi obtida por meio da Comunidade Acadêmica Federada (CAFe) no Portal de Periódicos da CAPES ou do *Google Scholar*, apenas dois artigos não puderam ser obtidos e, portanto, foram excluídos. Então, pela leitura do texto, buscamos identificar a) o problema a ser tratado, b) a forma de tratamento e, c) a técnica utilizada em cada artigo. Artigos que não apresentaram avaliação, simulada ou real, foram descartados. Também foram desconsideradas as revisões de literatura, artigos que não abordam otimização de datacenters ou que não apresentaram a classe do algoritmo utilizado. Com isso, o número de artigos passou a 122.

Classificamos então os artigos em categorias, de acordo com o objeto de cada um. Encontramos 25 artigos cujo principal problema é o Agendamento. A categoria seguinte denominamos otimizações de rede, com 22 artigos que se encaixam neste tema. E, por fim, 41 artigos classificamos como otimizações do uso de recursos físicos. Os demais artigos, totalizando 34, ou não puderam ser classificados em uma única classe principal ou tratavam de assuntos menos comuns e foram excluídos do estudo.

4. Resultados e discussão

Tabela 3: Otimizações de Agendamento

Artigo	Problema/Objetivo	Abordagem/Técnica
(35)	Application Scheduling	Global League Championship Algorithm (GBLCA)
(36)	Cloud Scheduling	Checkpoint League Championship Algorithm (CPLCA)
(37)	File assignement	MinCPP, DMinCPP
(38)	Flow scheduling/ Eficiência energética	Greedy based
(39)	Job Scheduling/Eficiência Energética	Integer-LP/GA, Local search
(40)	Multi-objective scheduling	Genetic Algorithm
(41)	Task Scheduling	ACO
(42)	Task Scheduling	Artificial Neural Network (ANN), GA
(43)	Task Scheduling	Bin Packing/Brokering algorithm
(44)	Task Scheduling	Combina min-min, min-max e *earliest deadline first*
(45)	Task Scheduling	Critical path/GA
(46)	Task Scheduling	Discrete Symbiotic Organism Search (DSOS)
(47)	Task Scheduling	FCFS, MCT, MET, Max/min, min/min, Sufferage
(48)	Task scheduling	GA, PSO, SA
(49)	Task scheduling	Integer-LP/Evolutionary Algorithm (EA)
(50)	Task Scheduling	Interval Scheduling/ Deadline Constrained Based Dynamic Load Balancing
(51)	Task scheduling	Min-Max Game
(52)	Task Scheduling	Ordinal Optimization (OO)
(53)	Task Scheduling	Partitioned Bin packing, Harmonic heuristic
(54)	Task scheduling/ Eficiência energética	Critical Path/Greedy based
(55)	Task scheduling/Eficiência energética	Compara 20 algoritmos, incluindo greedy based, Max/min, min/min, suffrage

(56)	Task scheduling/Eficiência energética	GA (NSGA-II, ev-MOGA), greedy
(57)	Task Scheduling/Eficiência Energética	Heterogeneous-Earliest-Finish-Time (HEFT) com DVFS
(58)	Workflow scheduling	Heterogeneous Earliest Finish Time (HEFT)
(59)	Workflow Scheduling	Proportional Deadline Constrained (PDC)/Deadline Constrained Critical Path (DCCP)

Tabela 4: Otimizações de Rede

Artigo	Problema/Objetivo	Abordagem/Técnica
(60)	Network flow/ Eficiência Energética	Quadratic assignment problem/ Greedy bin packing
(61)	Latency reduction	Lagrangian Relaxation
(62)	Network Function Virtualization (NFV)	Integer-LP, GA
(63)	Network Function Virtualization (NFV)	Combinatorial Optimization/ Markov algorithm
(64)	Network Function Virtualization (NFV)	Integer-LP/Propõem esquemas heurísticos SFC-RS e SFC-OS
(65)	Network Function Virtualization (NFV)	Integer-LP/Affinity-based Allocation (greedy)
(66)	Virtual Network Embedding (VNE)	MINTED, Tabu Search
(67)	Virtual Network Embedding (VNE)	Spanning Tree, Graph embedding/Multiple Exact Algorithms, Min-Max
(68)	Virtual Network Embedding (VNE)	Hungarian Algorithm
(69)	Virtual Network Mapping	Integer-LP/heurística com relaxamento de restrições
(70)	Network Flow, VM Consolidation	Greedy hierarchical
(71)	VN Embedding, VM Consolidation	Integer-LP/ Evolving VDC Provisioning across Federated data centers (EVPF)
(72)	VM Placement/Route Assignement	Non-convex quadratic constraint programming /Perturbation Algorithm
(73)	VN Embedding	Topological optimization/ Greedy Based
(74)	VN Embedding	Multiple Objectives LP/AI abstraction (Blocking Island)
(75)	VN Embedding	Artificial Immune System
(76)	VN Embedding	ACO
(77)	VN Embedding/Segurança	SA
(78)	VN Embedding/Segurança	GA
(79)	VNF, NFV, VM Consolidation	Graph Partitioning Problem/Game Theory

(80)	Network Flow	Topology-aware optimal RDT construction
(81)	Network Flow	k-Similar Greedy Tree (KSGT)

Tabela 5: Otimizações de Recursos Físicos

Artigo	Problema/Objetivo	Abordagem/Técnica
(82)	Capacity Allocation	MILP
(83)	Resource Provisioning, VM consolidation	Bin Packing/ FFD, statistics
(84)	Server Consolidation	Bin Packing/Integer-LP, BF
(85)	VM Allocation	Genetic Algorithm (GA)/Mixed Integer Linear Programming (MILP)
(86)	VM assignement/ Eficiência energética	Local search algorithm
(87)	VM Consolidation	Vector Bin Packing, GA; Permutation Pack (PP), First Fit (FF), First Fit Decreasing (FFD)
(88)	VM Consolidation	Bin Packing/ First-Fit (FF), Best-Fit (BF), First-Fit- Decreasing (FFD)
(89)	VM Consolidation	Híbrido Greedy, FFD, FF, BF, BFD, Round Robin
(90)	VM Consolidation	ACO
(91)	VM Consolidation	LP/RFAware, híbrida greedy
(92)	VM Consolidation	ABC
(93)	VM Consolidation	Firefly optimization algorithm (FFO)
(94)	VM Consolidation	Bayesian network
(95)	VM Consolidation	ACO
(96)	VM Consolidation	Memetic algorithm (Based on Evolutionary algorithm + local optimization)
(97)	VM Consolidation	Shuffled Frog-leaping Algorithm (SFLA)
(98)	VM Consolidation	First Search (DFS), FFD
(99)	VM Consolidation	Local search
(100)	VM Consolidation e VM Migration	Integer-LP, Knapsack problem/CPLEX, BFD
(101)	VM Consolidation e VM Migration	GA e ABC
(102)	VM Consolidation e VM Migration	A*
(70)	VM Consolidation, Network Flow	Greedy hierarchical
(71)	VM Consolidation, VN Embedding	Integer-LP/ Evolving VDC Provisioning across Federated data centers (EVPF)

(103)	VM Consolidation/ Eficiência energética	Ant Colony System
(104)	VM consolidation/ Eficiência energética	AS
(105)	VM Consolidation/ Eficiência energética	ARIMA (media móvel)
(106)	VM Consolidation/ Eficiência energética	PSO
(107)	VM Consolidation/ Eficiência energética	GA
(108)	VM Consolidation/ Eficiência energética	Evolutionary Game Theory
(109)	VM Consolidation/ Eficiência energética	Least-Reliable-First(LRF) and Decreased-Density-Greedy(DDG)
(2)	VM Consolidation/ Eficiência energética	Biogeography-based optimization (BBO)
(110)	VM Consolidation/ Eficiência energética	GA e ANN
(111)	VM Consolidation/ Eficiência energética	Adaptive Three-threshold Framework (baseado em políticas)
(112)	VM Consolidation/ Eficiência energética	Combina métodos estatísticos e políticas
(113)	VM Consolidation/ Eficiência energética	Integer-LP/Feasibility Driven StochasticVM Placement (FDSP)
(114)	VM Consolidation/ Eficiência energética	Vector Bin Packing/ Compara FFDprod, dotproduct e norm2
(15)	VM Consolidation/Green DC	Integer-LP/ Tree and Forest algorithm, BF
(115)	VM Placement	Hungarian Algorithm, Max 3-dimensional matching, SA, Greedy
(116)	VM Placement	Quadratic Assignment Problem/Greedy-based
(117)	VM Placement	Constraint Programming, Greedy Algorithm
(72)	VM Placement/Route Assignement	Non-convex quadratic constraint programming /Perturbation Algorithm

Dentre os artigos dedicados a problemas de agendamento (*Tab. 3*), 23 utilizaram algum método heurístico (92%) e 2 fizeram uso da Teoria dos Jogos (8%) e 1 utilizou Machine Learning (4%) mas também utilizou meta-heurística. O uso de métodos meta-heurísticos foi feito por 12 artigos (48%), sendo que 6 destes usaram métodos baseados em metáforas (24%).

Dentre os artigos dedicados a problemas de rede (*Tab. 4*), 16 utilizaram métodos heurísticos (72%), 6 destes meta-heurísticos (27%), dos quais apenas 2 são baseados em metáforas (9%). Encontramos 5 artigos que utilizaram métodos matemáticos tradicionais (23%), o que pode apontar uma melhor tratabilidade dos problemas desta categoria (*Gráfico 1*).

Na categoria dedicada a otimização de recursos físicos (*Tab. 5*), 36 utilizam de algum método heurístico (88%), dos quais 18 fazem uso de meta-heurística (44%), onde 9 são baseadas em metáforas (22%). Quatro artigos fazem uso de métodos estatísticos (10%), 1 utiliza Machine Learning (2%) mas também utiliza meta-heurística e a Teoria dos jogos

é utilizada por 1. Encontramos 5 artigos que realizam uma abordagem com métodos que chamamos tradicionais (12%), e um deles incorpora estatística.

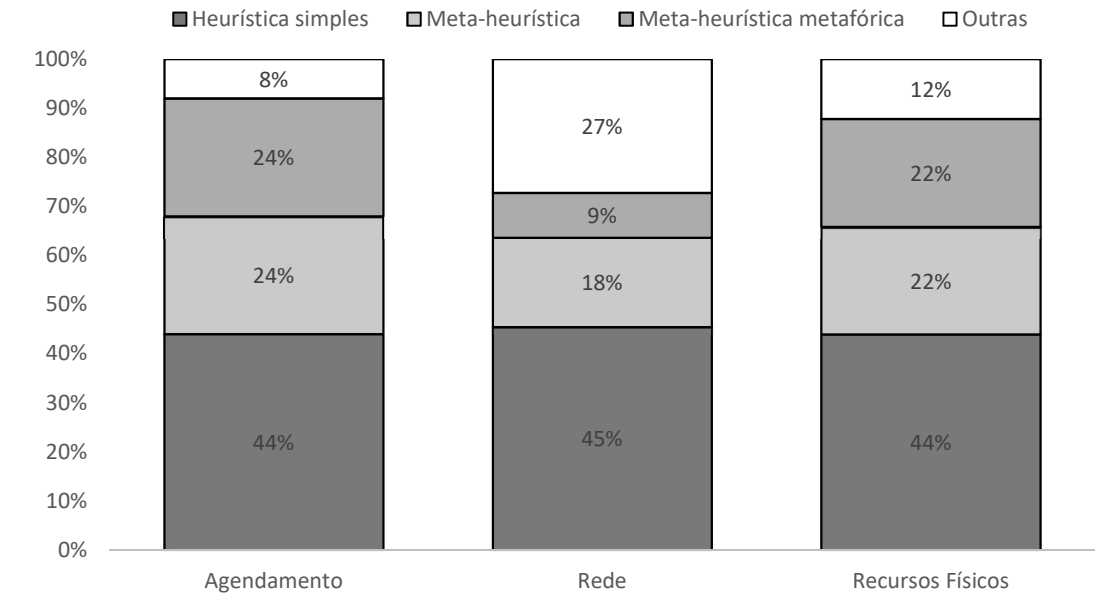

Grafico 1: Comparativo do Uso de Técnicas de Otimização

5. Conclusão

O crescimento do volume de dados que se operam atualmente em uma perspectiva global, assim como o poder computacional subjacente necessário para trabalhar estas informações, coloca o datacenter em uma posição central para o desenvolvimento econômico, tecnológico e cultural. A redução dos custos operacionais assim como a redução, cada vez mais necessária, da pegada ambiental força-nos a procurar meios de operar de forma mais eficiente. Neste trabalho elencamos as técnicas mais recentes e preparamos uma bibliografia que auxilie o projeto de futuros datacenters, assim como a reestruturação dos atualmente operantes. O levantamento aqui realizado indica uma direção para o desenvolvimento de novas abordagens que atendam as demandas dinâmicas e complexas de um datacenter que presta serviços de processamento e armazenamento na nuvem.

Algoritmos heurísticos são um campo de pesquisa bastante recente e tem crescido tanto em quantidade de abordagens quanto em variedade. O uso de *Machine Learning*, que tem remodelado tantas áreas da atividade humana recentemente, ainda é tímido no datacenter e caberia melhor explorar a usabilidade destas técnicas. Fica bastante evidente que o principal foco dos trabalhos em otimização é a relação entre o recurso virtualizado e sua contraparte física. E, justamente, neste espaço heurísticas mais sofisticadas tornam-se necessárias.

Limitações necessárias de escopo e do tempo determinado para o desenvolvimento da pesquisa nos obrigaram a explorar apenas uma pequena faixa dos aspectos que podem ser

otimizados em um datacenter. Estudos adicionais são necessários para validar os resultados aqui encontrados.

6. REFERÊNCIAS

1 Varasteh, A.; Goudarzi, M. Server Consolidation Techniques in Virtualized Data Centers: A Survey. **IEEE Systems Journal**, v. 11, n. 2, p. 772–783, 2017.

2 Zheng, Q.; Li, R.; et al. Virtual machine consolidated placement based on multi-objective biogeography-based optimization. **Future Generation Computer Systems**, v. 54, p. 95–122, 2016.

3 Mann, Z.Á. Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms. **ACM Computing Surveys**, v. 48, n. 1, p. 1–34, 2015.

4 Usmani, Z.; Singh, S. A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. **Physics Procedia**, v. 78, p. 491–498, 2016.

5 Alkhanak, E.N.; Lee, S.P.; et al. Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. **Journal of Systems and Software**, v. 113, p. 1–26, 2016.

6 Choudhary, A.; Rana, S.; et al. A Critical Analysis of Energy Efficient Virtual Machine Placement Techniques and its Optimization in a Cloud Computing Environment. **Physics Procedia**, v. 78, p. 132–138, 2016.

7 Agrawal, B.M.; Kayal, N.; et al. PRIMES is in P. **Annals of Mathematics**, v. 160, n. 2, p. 781–793, 2004.

8 Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. **Proceedings of the London Mathematical Society**, v. 42, n. 2, p. 230–265, 1936.

9 Petzold, C. **The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine**. Indianapolis: John Wiley & Sons, 2008.

10 Gödel, K.; Church, A.; et al. **The undecidable: Basic papers on undecidable propositions, unsolvable problems and computable functions**. New York: Raven Press, 1965.

11 Soare, R.I. **Recursively enumerable sets and degrees: A study of computable functions and computably generated sets**. New York: Springer-Verlag, 1987.

12 Aaronson, S. Why Philosophers Should Care About Computational Complexity. In: JB Copeland (Ed.); **Gödel, Turing, Church, and beyond**. Cambridge, MA: MIT Press, 2013, p. 261.

13 Singh, P.; Dutta, M.; et al. A review of task scheduling based on meta-heuristics approach in cloud computing. **Knowledge and Information Systems**, v. 52, n. 1, p. 1–51, 2017.

14 Shabeera, T.P.; Madhu Kumar, S.D.; et al. Curtailing job completion time in MapReduce clouds through improved Virtual Machine allocation. **Computers and Electrical Engineering**, v. 58, p. 190–202, 2017.

15 Tseng, F.H.; Chen, C.Y.; et al. Service-Oriented Virtual Machine Placement Optimization for Green Data Center. **Mobile Networks and Applications**, v. 20, n. 5, p. 556–566, 2015.

16 Karp, R.M. Reducibility among combinatorial problems. In: **50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art**. 2010, p. 219–241.

17 Schrijver, A. On the History of Combinatorial Optimization. **Handbooks in Operations Research and Management Science: Discrete Optimization**, v. 12, n. Till 1960, p. 1–57, 2005.

18 Fortnow, L.; Homer, S. A Short History of Computational Complexity. **Science**, v. 80, p. 1–26, 2002.

19 Karp, R.M.; Lipton, R.J. Turing machines that take advice. **L'Ensegnement Mathématique**, v. 28, p. 191–209, 1982.

20 Ascó, A.; Atkin, J.A.D.; et al. An analysis of constructive algorithms for the airport baggage sorting station assignment problem. **Journal of Scheduling**, v. 17, n. 6, p. 601–619, 2014.

21 Ghosh, A. A well organized energy efficient cloud data center using simulated annealing optimization technique. **International Journal of Advanced Research in Computer Science**, v. 8, n. 7, 2017.

22 Larumbe, F.; Sanso, B. A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks. **IEEE Transactions on Cloud Computing**, v. 1, n. 1, p. 22–35, 2013.

23 Masdari, M.; Salehi, F.; et al. A Survey of PSO-Based Scheduling Algorithms in Cloud Computing. **Journal of Network and Systems Management**, v. 25, n. 1, p. 122–158, 2017.

24 Elbeltagi, E.; Hegazy, T.; et al. Comparison among five evolutionary-based optimization algorithms.

25 Dorigo, M.; Maniezzo, V.; et al. Ant System: Optimization by a Colony of Cooperating Agents. **IEEE Transactions on Systems, Man, And Cybernetics - Part B (Cybernetics)**, v. 26, n. 1, p. 1–13, 1996.

26 Dorigo, M.; Stützle, T. **Optimization**.

27 Karaboga, D.; Ozturk, C. A novel clustering approach: Artificial Bee Colony (ABC) algorithm. **Applied Soft Computing Journal**, v. 11, n. 1, p. 652–657, 2011.

28 Pham, D.T.; Castellani, M. The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems. **Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science**, v. 223, n. 12, p. 2919–2938, 2009.

29 Eusuff, M.M.; Lansey, K.E. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. **Journal of Water Resources Planning and Management**, v. 129, n. 3, p. 210–225, 2003.

30 Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. **Computer Methods in Applied Mechanics and Engineering**, v. 194, n. 36–38, p. 3902–3933, 2005.

31 Lucas, C.; Nasiri-Gheidari, Z.; et al. Application of an imperialist competitive algorithm to the design of a linear induction motor. **Energy Conversion and Management**, v. 51, n. 7, p. 1407–1411, 2010.

32 Yang, X.-S. A New Metaheuristic Bat-Inspired Algorithm. In: EJRG et Al. (Ed.); **Nature Inspired Cooperative Strategies for Optimization (NISCO 2010)**. Berlin: Springer Berlin, v.SCI 284.2010, p. 65–74.

33 Yang, X. **Nature-Inspired Metaheuristic Algorithms Second Edition**. Frome, UK: Luniver Press, 2010.

34 Li, X.X.; Shao, Z.; et al. An optimizing method based on autonomous animals: fish-swarm algorithm. **Syst Eng Theory Pratice**, v. 22, n. 11, p. 32–38, 2002.

35 Abdulhamid, S.M.; Abd Latiff, M.S.; et al. Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. **PLoS ONE**, v. 11, n. 7, p. 1–18, 2016.

36 Abdulhamid, S.M.; Latiff, M.S.A. A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness. **Applied Soft Computing Journal**, v. 61, p. 670–680, 2017.

37 Dong, B.; Li, X.; et al. Towards minimizing disk I/O contention: A partitioned file assignment approach. **Future Generation Computer Systems**, v. 37, p. 178–190, 2014.

38 Li, D.; Yu, Y.; et al. Willow: Saving Data Center Network Energy for Network-Limited Flows. **IEEE Transactions on Parallel and Distributed Systems**, v. 26, n. 9, p. 2610–2620, 2015.

39 Wang, X.; Wang, Y.; et al. A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing. **Future Generation Computer Systems**, v. 36, p. 91–101, 2014.

40 Frîncu, M.E.; Frincu, M.E. Scheduling highly available applications on cloud environments. **Future Generation Computer Systems**, v. 32, n. 1, p. 138–153, 2014.

41 Moon, Y.; Yu, H.; et al. A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. **Human-centric Computing and Information Sciences**, v. 7, n. 1, p. 28, 2017.

42 Grzonka, D.; Kolodziej, J.; et al. Artificial Neural Network support to monitoring of the evolutionary driven security aware scheduling in computational distributed environments. **Future Generation Computer Systems**, v. 51, p. 72–86, 2015.

43 Bassem, C.; Bestavros, A. Multi-Capacity Bin Packing with Dependent Items and

its Application to the Packing of Brokered Workloads in Virtualized Environments. **Future Generation Computer Systems**, v. 72, p. 129–144, 2017.

44 Vasiliu, L.; Pop, F.; et al. A Hybrid Scheduler for Many Task Computing in Big Data Systems. **International Journal of Applied Mathematics and Computer Science**, v. 27, n. 2, p. 385–399, 2017.

45 Wang, J.; Taal, A.; et al. Planning virtual infrastructures for time critical applications with multiple deadline constraints. **Future Generation Computer Systems**, v. 75, p. 365–375, 2017.

46 Abdullahi, M.; Ngadi, M.A.; et al. Symbiotic Organism Search optimization based task scheduling in cloud computing environment. **Future Generation Computer Systems**, v. 56, p. 640–650, 2016.

47 Madni, S.H.H.; Abd Latiff, M.S.; et al. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. **PLoS ONE**, v. 12, n. 5, p. 1–26, 2017.

48 Shishido, H.Y.; Estrella, J.C.J.C.; et al. Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. **Computers and Electrical Engineering**, 2017.

49 Teylo, L.; Paula, U. de; et al. A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. **Future Generation Computer Systems**, v. 76, p. 1–17, 2017.

50 Kumar, M.; Sharma, S.C. Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. **Computers and Electrical Engineering**, 2017.

51 Koutsandria, G.; Skevakis, E.; et al. Can everybody be happy in the cloud? Delay, profit and energy-efficient scheduling for cloud services. **Journal of Parallel and Distributed Computing**, v. 96, p. 202–217, 2016.

52 Zhang, F.; Cao, J.; et al. Multi-objective scheduling of many tasks in cloud platforms. **Future Generation Computer Systems**, v. 37, p. 309–320, 2014.

53 Fan, M.; Han, Q.; et al. Enhanced fixed-priority real-time scheduling on multi-core platforms by exploiting task period relationship. **Journal of Systems and Software**, v. 99, p. 85–96, 2015.

54 Li, X.; Cai, Z. Elastic Resource Provisioning for Cloud Workflow Applications. **IEEE Transactions on Automation Science and Engineering**, v. 14, n. 2, p. 1195–1210, 2017.

55 Nesmachnow, S.; Dorronsoro, B.B.; et al. Energy-Aware Scheduling on Multicore Heterogeneous Grid Computing Systems. **Journal of Grid Computing**, v. 11, n. 4, p. 653–680, 2013.

56 Iturriaga, S.; Nesmachnow, S. Scheduling Energy Efficient Data Centers Using Renewable Energy. **Electronics**, v. 5, n. 4, p. 71, 2016.

57 Tang, Z.; Qi, L.; et al. An Energy-Efficient Task Scheduling Algorithm in DVFS-

enabled Cloud Environment. **Journal of Grid Computing**, v. 14, n. 1, p. 55–74, 2016.

58 Durillo, J.J.; Prodan, R.; et al. Pareto tradeoff scheduling of workflows on federated commercial Clouds. **Simulation Modelling Practice and Theory**, v. 58, n. February, p. 95–111, 2015.

59 Arabnejad, V.; Bubendorfer, K.; et al. Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. **Future Generation Computer Systems**, v. 75, p. 348–364, 2017.

60 Fang, W.; Liang, X.; et al. VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. **Computer Networks**, v. 57, n. 1, p. 179–196, 2013.

61 Amiri, M.; Osman, H. Al; et al. Toward Delay-Efficient Game-Aware Data Centers for Cloud Gaming. **ACM Transactions on Multimedia Computing, Communications, and Applications**, v. 12, n. 5s, p. 1–19, 2016.

62 Rankothge, W.; Le, F.; et al. Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms. **IEEE Transactions on Network and Service Management**, v. 14, n. 2, p. 343–356, 2017.

63 Wang, P.; Lan, J.; et al. Dynamic function composition for network service chain: Model and optimization. **Computer Networks**, v. 92, p. 408–418, 2015.

64 Yi, B.; Wang, X.; et al. Design and evaluation of schemes for provisioning service function chain with function scalability. **Journal of Network and Computer Applications**, v. 93, p. 197–214, 2017.

65 Bhamare, D.; Samaka, M.; et al. Optimal virtual network function placement in multi-cloud service function chaining architecture. **Computer Communications**, v. 102, p. 1–16, 2017.

66 Ayoubi, S.; Assi, C.; et al. MINTED: Multicast virtual network embedding in cloud data centers with delay constraints. **IEEE Transactions on Communications**, v. 63, n. 4, p. 1291–1305, 2015.

67 Fuerst, C.; Pacut, M.; et al. Data locality and replica aware virtual cluster embeddings. **Theoretical Computer Science**, v. 697, p. 37–57, 2017.

68 Ghalami Osgouei, A.; Khorsandi Koohanestani, A.; et al. Online assignment of non-SDN virtual network nodes to a physical SDN. **Computer Networks**, v. 129, p. 105–116, 2017.

69 Sun, G.; Liao, D.; et al. Towards provisioning hybrid virtual networks in federated cloud data centers. **Future Generation Computer Systems**, 2017.

70 Roh, H.; Jung, C.; et al. Joint flow and virtual machine placement in hybrid cloud data centers. **Journal of Network and Computer Applications**, v. 85, p. 4–13, 2017.

71 Sun, G.; Liao, D.; et al. The efficient framework and algorithm for provisioning evolving VDC in federated data centers. **Future Generation Computer Systems**, v. 73, p. 79–89, 2017.

72 Yan, F.; Lee, T.T.; et al. Congestion-Aware Embedding of Heterogeneous Bandwidth Virtual Data Centers with Hose Model Abstraction. **IEEE/ACM Transactions on Networking**, v. 25, n. 2, p. 806–819, 2017.

73 Ma, J. Resource management framework for virtual data center embedding based on software defined networking. **Computers and Electrical Engineering**, v. 60, p. 76–89, 2017.

74 Wang, T.; Hamdi, M. Presto: Towards efficient online virtual network embedding in virtualized cloud data centers. **Computer Networks**, v. 106, p. 196–208, 2016.

75 Zhang, Z.; Su, S.; et al. Adaptive multi-objective artificial immune system based virtual network embedding. **Journal of Network and Computer Applications**, v. 53, p. 140–155, 2015.

76 Fajjari, I.; Aitsaadi, N.; et al. A new virtual network static embedding strategy within the Cloud's private backbone network. **Computer Networks**, v. 62, p. 69–88, 2014.

77 Oliveira, R.R.; Marcon, D.S.; et al. Opportunistic resilience embedding (ORE): Toward cost-efficient resilient virtual networks. **Computer Networks**, v. 89, p. 59–77, 2015.

78 Pathak, I.; Vidyarthi, D.P.P. A model for virtual network embedding across multiple infrastructure providers using genetic algorithm. **Science China Information Sciences**, v. 60, n. 4, 2017.

79 Leivadeas, A.; Kesidis, G.; et al. A graph partitioning game theoretical approach for the VNF service chaining problem. **IEEE Transactions on Network and Service Management**, v. 14, n. 4, p. 1, 2017.

80 Chen, W.; Paik, I.; et al. Tology-Aware Optimal Data Placement Algorithm for Network Traffic Optimization. **IEEE Transactions on Computers**, v. 65, n. 8, p. 2603–2617, 2016.

81 Jia, X.; Li, Q.; et al. A low overhead flow-holding algorithm in software-defined networks. **Computer Networks**, v. 124, p. 170–180, 2017.

82 Ciavotta, M.; Ardagna, D.; et al. A mixed integer linear programming optimization approach for multi-cloud capacity allocation. **Journal of Systems and Software**, v. 123, p. 64–78, 2017.

83 Song, W.; Xiao, Z.; et al. Adaptive resource provisioning for the cloud using online bin packing. **IEEE Transactions on Computers**, v. 63, n. 11, p. 2647–2660, 2014.

84 Mazumdar, S.; Pranzo, M. Power efficient server consolidation for Cloud data center. **Future Generation Computer Systems**, v. 70, p. 4–16, 2017.

85 Guerout, T.; Gaoua, Y.; et al. Mixed integer linear programming for quality of service optimization in Clouds. **Future Generation Computer Systems**, v. 71, p. 1–17, 2017.

86 Kessaci, Y.; Melab, N.; et al. A multi-start local search heuristic for an energy

efficient VMs assignment on top of the OpenNebula cloud manager. **Future Generation Computer Systems**, v. 36, p. 237–256, 2014.

87 Hallawi, H.; Mehnen, J. orn; et al. Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation. **Future Generation Computer Systems**, v. 69, p. 1–10, 2017.

88 Mann, Z.A.Z.Á.Z.A. Resource optimization across the cloud stack. **IEEE Transactions on Parallel and Distributed Systems**, v. 29, n. 1, p. 169–182, 2017.

89 Rahman, M.M.; Graham, P. Responsive and efficient provisioning for multimedia applications. **Computers and Electrical Engineering**, v. 53, p. 458–468, 2016.

90 Sun, X.; Zhang, K.; et al. Multi-Population Ant Colony Algorithm for Virtual Machine Deployment. **IEEE Access**, v. 5, p. 27014–27022, 2017.

91 Sunil Rao, K.; Santhi Thilagam, P. Heuristics based server consolidation with residual resource defragmentation in cloud data centers. **Future Generation Computer Systems**, v. 50, p. 87–98, 2015.

92 Jiang, J.; Feng, Y.; et al. DataABC: A fast ABC based energy-efficient live VM consolidation policy with data-intensive energy evaluation model. **Future Generation Computer Systems**, v. 74, p. 132–141, 2017.

93 Kansal, N.J.; Chana, I. Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach. **Journal of Grid Computing**, v. 14, n. 2, p. 327–345, 2016.

94 Li, Z.; Yan, C.; et al. Bayesian network-based Virtual Machines consolidation method. **Future Generation Computer Systems**, v. 69, p. 75–87, 2017.

95 Liu, X.-F.; Zhan, Z.-H.; et al. An Energy Aware Unified Ant Colony System for Dynamic Virtual Machine Placement in Cloud Computing. **Energies**, v. 10, n. 5, p. 609, 2017.

96 Lopez-Pires, F.; Baran, B.; et al. Many-Objective Virtual Machine Placement. **Journal of Grid Computing**, v. 15, n. 2, p. 161–176, 2017.

97 Luo, J.P.; Li, X.; et al. Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. **Expert Systems with Applications**, v. 41, n. 13, p. 5804–5816, 2014.

98 Xu, J.; Tang, J.; et al. Enhancing survivability in virtualized data centers: A service-aware approach. **IEEE Journal on Selected Areas in Communications**, v. 31, n. 12, p. 2610–2619, 2013.

99 Zeng, L.; Wang, Y. Optimization on content service with local search in cloud of clouds. **Journal of Network and Computer Applications**, v. 40, n. 1, p. 206–215, 2014.

100 Wang, X.; Chen, X.; et al. Delay-cost tradeoff for virtual machine migration in cloud data centers. **Journal of Network and Computer Applications**, v. 78, p. 62–72, 2017.

101 Zhang, W.; Han, S.; et al. Network-aware virtual machine migration in an overcommitted cloud. **Future Generation Computer Systems**, v. 76, p. 428–442, 2017.

102 Dow, E.M.; Matthews, J.N. WAYFINDER: parallel virtual machine reallocation through A* search. **Memetic Computing**, v. 8, n. 4, p. 255–267, 2016.

103 Farahnakian, F.; Ashraf, A.; et al. Using Ant Colony System to Consolidate VMs for Green Cloud Computing. **IEEE Transactions on Services Computing**, v. 8, n. 2, p. 187–198, 2015.

104 Ghosh, R.; Longo, F.; et al. Stochastic model driven capacity planning for an infrastructure-as-a-service cloud. **IEEE Transactions on Services Computing**, v. 7, n. 4, p. 667–680, 2014.

105 Vakilinia, S.; Heidarpour, B.; et al. Energy efficient resource allocation in cloud computing environments. **IEEE Access**, v. 4, p. 8544–8557, 2016.

106 Wang, S.; Zhou, A.; et al. Provision of Data-Intensive Services Through Energy- and QoS-Aware Virtual Machine Placement in National Cloud Data Centers. **IEEE Transactions on Emerging Topics in Computing**, v. 4, n. 2, p. 290–300, 2016.

107 Wen, Y.; Li, Z.; et al. Energy-Efficient Virtual Resource Dynamic Integration Method in Cloud Computing. **IEEE Access**, v. 5, p. 12214–12223, 2017.

108 Xiao, Z.; Jiang, J.; et al. A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory. **Journal of Systems and Software**, v. 101, p. 260–272, 2015.

109 Zhao, L.; Lu, L.; et al. Online virtual machine placement for increasing cloud provider's revenue. **IEEE Transactions on Services Computing**, v. 10, n. 2, p. 273–285, 2017.

110 Zheng, S.; Zhu, G.; et al. Towards an adaptive human-centric computing resource management framework based on resource prediction and multi-objective genetic algorithm. **Multimedia Tools and Applications**, v. 76, n. 17, p. 17821–17838, 2017.

111 Zhou, Z.; Abawajy, J.; et al. Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms. **Future Generation Computer Systems**, 2017.

112 Zhu, W.; Zhuang, Y.; et al. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. **Future Generation Computer Systems**, v. 69, p. 66–74, 2017.

113 Chen, Y.; Chen, X.; et al. Stochastic scheduling for variation-aware virtual machine placement in a cloud computing CPS. **Future Generation Computer Systems**, 2017.

114 Gaggero, M.; Caviglione, L. Predictive Control for Energy-Aware Consolidation in Cloud Datacenters. **IEEE Transactions on Control Systems Technology**, v. 24, n. 2, p. 461–474, 2016.

115 Deng, H.; Huang, L.; et al. Optimizing virtual machine placement in distributed

clouds with M/M/1 servers. **Computer Communications**, v. 102, p. 107–119, 2017.

116 Ilkhechi, A.R.; Korpeoglu, I.; et al. Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components. **Computer Networks**, v. 91, p. 508–527, 2015.

117 Mann, Z.A.Z.Á.Z.A. Multicore-Aware Virtual Machine Placement in Cloud Data Centers. **IEEE Transactions on Computers**, v. 65, n. 11, p. 3357–3369, 2016.