Aplicação das técnicas de Otimização por Enxame de Partículas e Path Relinking para o problema de alocação de contêineres em centro de dados

João Paulo de Araújo[1], **Filipe de Matos** [2], **Fernando Antonio Mota Trinta** [1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici – Bloco 902 – CEP 60.440-900 – Fortaleza – CE – Brasil

[2]GOHaN Research Group – Universidade Federal do Ceará (UFC)
Campus de Crateús – CEP 63700-00 – Crateús – CE – Brasil

joaopauloaraujocjpa@alu.ufc.br, filipe.fernandes@crateus.ufc.br, fernando.trinta@dc.ufc.br

Resumo. *A virtualização por contêineres destaca-se como uma forma mais leve de virtualização, que permite o provisionamento rápido de serviços, bem como a portabilidade destes. Devido à fatores como heterogeneidade na configuração dos contêineres e à dimensionalidade dos centros de dados hospedeiros, determinar uma alocação ótima configura-se como um problema combinatório difícil, pois trata-se de um problema, na maioria dos casos, com um amplo espaço de busca. Diante disto, o presente trabalho apresenta uma política de alocação de contêineres denominada PSOPR, que tem como base a técnica de Otimização por Enxame de Partículas em conjunto com a meta-heurística Path Relinking, buscando a consolidação de centros de dados sem que haja impactos desfavoráveis ao desempenho das aplicações. Utilizando o CloudSim como ferramenta de simulação, e com base nas métricas de consumo de energia, violação de SLA e quantidade de máquinas virtuais utilizadas, os resultados dos experimentos indicaram que a política PSOPR consumiu, em média, 25.38% e 24.61% menos energia que as políticas First-Come, First-Served (FCFS) e Aleatória (Random), respectivamente. Além disso, ela também mostrou bons resultados em termos de nível de violação de SLA. Observou-se que, em média, a política proposta violou a SLA em 10%, um dos melhores resultados dentre as políticas avaliadas.*

Abstract. *Container virtualization stands out as a lighter form of virtualization, enabling rapid service provisioning and portability. Due to factors such as heterogeneity in container configurations and the dimensionality of hosting data centers, determining an optimal allocation poses a challenging combinatorial problem, often involving a wide search space. In this context, this study introduces a container allocation policy named PSOPR, based on the Particle Swarm Optimization technique combined with the metaheuristic Path Relinking. The goal is to consolidate data centers without adversely affecting application performance. Using CloudSim as a simulation tool and relying on metrics such as energy consumption, SLA violation, and the number of virtual machines used, experimental results indicated that the PSOPR policy consumed, on average,*

25.38% and 24.61% less energy than the First-Come, First-Served (FCFS) and Random policies, respectively. Furthermore, it demonstrated favorable results in terms of SLA violation levels, with an average violation of 10%, one of the best outcomes among the evaluated policies.

1. Introdução

A computação em nuvem tornou-se uma das tecnologias mais predominantes nos últimos anos, fornecendo serviços de computação sob demanda pela Internet para economias, sociedade e indivíduos [Varghese and Buyya 2018]. A demanda pelo modelo é impulsionada pelo crescente aumento de cargas de trabalho dos mais diversos tipos de serviços, como, por exemplo, serviços de processamento de dados, *streaming* de áudio/vídeo e armazenamento em nuvem, o que tornam maiores as expectativas para uma adesão ainda mais considerável ao modelo [Ahmad et al. 2021].

A virtualização é uma tecnologia fundamental no cenário atual do paradigma de computação em nuvem [Zhang et al. 2010]. Ela desempenha um papel primordial na otimização do uso dos recursos computacionais e energéticos dos provedores de serviços, permitindo a implantação de diversas aplicações em uma única máquina física, sem que haja interferência entre elas. Tudo isso se deve ao fato de que a virtualização abstrai os recursos físicos de computação em recursos virtuais. Dentre os tipos existentes, a virtualização no nível de sistemas operacionais, também conhecida como virtualização por contêineres, surge como uma tecnologia promissora no fornecimento de serviços em nuvem devido à sua implantação flexível, portabilidade e escalabilidade, especialmente em microsserviços [Ahmad et al. 2021]. Além disso, a virtualização baseada em contêineres é capaz de simplificar processos como instalação e lançamento de serviços [de Assuncao et al. 2018].

A política de alocação de contêineres desempenha um papel crucial na implementação de virtualização por contêineres. Ela é a responsável por determinar o *nó*, sendo este físico ou virtual, onde cada contêiner é alocado em um Centro de Dados (em inglês *Data Center* - DC). No entanto, uma vez que diferentes contêineres podem possuir diferentes configurações (*e.g.*, capacidade de processamento, tamanho de memória RAM e capacidade de armazenamento), é necessário testar todas as possíveis combinações de alocação. O espaço de busca gerado neste processo pode ser muito amplo, visto que o número de combinações possíveis cresce exponencialmente. Consequentemente, o tempo de execução necessário para testar todas as combinações possíveis, tende a ser muito grande. Desta forma, por se tratar de um problema que envolve tomada de decisão em um cenário contendo uma ampla quantidade de possibilidades, a alocação de contêineres pode ser classificada como um problema de otimização combinatória. Além disso, problemas de otimização para os quais não se dispõe de uma solução conhecida em tempo polinomial são categorizados como problemas *NP-difíceis* [Papadimitriou and Steiglitz 1998], sendo este o contexto do problema em questão.

Nos casos em que buscar a solução exata seja inviável computacionalmente, principalmente devido ao tempo de resposta, abordagens que alcancem respostas com resultados aproximados são opções aceitáveis para tratar um dado problema. Dentre essas abordagens, a utilização de uma meta-heurística é uma boa alternativa, haja vista a possibilidade de se obter boas soluções com tempo de execução considerável, conforme a

rapidez da resposta que cada problema abordado necessita.

Nesse contexto, este trabalho apresenta uma política de alocação de contêineres que busca distribuir a maior quantidade de contêineres no menor número de servidores disponíveis em um DC. Para isso, o presente estudo explora a capacidade da metaheurística Otimização por Enxame de Partículas, (do inglês: *Particle Swarm Optimization* ou PSO), um método criado a partir da análise do comportamento social dos pássaros [Kennedy and Eberhart 1995], juntamente com o *Path Relinking*, uma metaheurística baseada em população capaz de gerar novas soluções a partir de duas outras, ditas solução inicial e solução guia, aplicando uma série de operações em suas iterações [Glover et al. 2000].

A hipótese de combinar o PSO com o *Path Relinking* é fundamentada com base na característica de que ambos operam com uma população de soluções. A partir das avaliações experimentais realizadas, a implementação desenvolvida no presente estudo obteve uma média de consumo energético $25,38\%$ menor em comparação com uma política tendo como base a abordagem *First-Come, First-Served* (FCFS), e $24,61\%$ menor em comparação com uma política desenvolvida com base na abordagem aleatória. Além disso, a política desenvolvida demonstrou, em média, uma violação de SLA de 10% com desvio padrão de 1%, sendo um dos melhores resultados obtidos dentre as políticas avaliadas.

O restante deste documento está assim organizado: A Seção 2 apresenta conceitos teóricos fundamentais para a compreensão deste estudo; a Seção 3 aborda os trabalhos relacionados, que embasam ainda mais a problemática discutida; na Seção 3, são apresentados os principais pontos da implementação desenvolvida; na Seção 4, a compilação dos experimentos é apresentada; por fim, a Seção 5 traz as considerações finais e aponta trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta os conceitos teóricos que servem de base para a compreensão de conceitos importantes citados no escopo do presente trabalho. Estes conceitos são cruciais para compreender não apenas o conteúdo deste estudo, mas também para interpretar seus resultados e considerações.

2.1. Otimização por Enxame de Partículas (PSO)

A meta-heurística Otimização por Enxame de Partículas (PSO) é um algoritmo de otimização que se baseia no comportamento coletivo de uma população, como o movimento coordenado de um enxame de pássaros ou cardumes. [Kennedy and Eberhart 1995]. Alguns termos fazem parte do contexto do PSO e são fundamentais para a compreensão do algoritmo:

- **Partícula:** É um indivíduo que compõe um enxame. Cada uma das partículas representa uma possível solução para o problema abordado;
- **Enxame ou População:** Conjunto de partículas;
- **Função *fitness*:** Responsável por atribuir um valor numérico para cada partícula com o intuito de avaliar quão boa é a posição desta em cada iteração;
- **Velocidade:** Determina a direção na qual ela se movimentará, visando melhorar sua posição atual;

- **Posição:** Solução representada pela partícula no espaço de busca;
- $pbest$**:** Melhor posição alcançada por uma partícula;
- $gbest$**:** Melhor posição alcançada em toda a população;
- **Fator de individualidade** $(c1)$**:** Influência do $gbest$ em relação ao $pbest$;
- **Fator de sociabilidade** $(c2)$**:** Exercer impacto na atração que a partícula possui sobre o $gbest$;
- **Coeficiente de inércia** (ω)**:** Responsável por controlar a influência dos valores anteriores da velocidade no cálculo da velocidade atual.

Ao explorar o espaço de busca com PSO, as partículas interagem dinamicamente, compartilhando informações sobre suas posições e desempenhos individuais. Essa colaboração possibilita a combinação da abordagem com outros métodos com capacidade de refinar as informações das partículas, como, por exemplo, o *Path Relinking*.

2.2. *Path Relinking*

O *Path Relinking* é a forma generalizada da abordagem evolutiva chamada *Scatter Search*, uma técnica utilizada para resolver problemas de otimização combinatória [Glover et al. 2000]. Sua principal característica é a habilidade de gerar novas soluções a partir de duas pré-existentes, comumente denominadas soluções "inicial" e "guia".

A geração dessas novas soluções ocorre por meio de uma sequência de movimentos que parte da solução inicial e gradualmente constrói um caminho em direção à solução guia, sendo este o motivo para o nome do método. Desta forma, um conjunto diversificado de soluções intermediárias é gerado, com o intuito de explorar o espaço de busca de maneira abrangente. Em alguns casos, o algoritmo é capaz de identificar uma solução superior às soluções inicial e guia, o que representa um sucesso na otimização do problema em que o *Path Relinking* é aplicado.

Dentre o conjunto de soluções geradas, o algoritmo pode encontrar uma ainda melhor que a inicial e guia. Por se basear em população, o *Path Relinking* pode ser combinado com outros algoritmos que também possuem essa característica, como, por exemplo, o PSO.

3. Trabalhos Relacionados

Em conformidade com o contexto do presente estudo, para aprofundar ainda mais o embasamento referente ao tema abordado, esta seção apresenta estudos presentes na literatura sobre o gerenciamento de contêineres. Os estudos apresentados foram selecionados considerando aspectos como questionamentos levantados e abordagens utilizadas.

[Ahmad et al. 2021] apresenta uma visão sobre técnicas existentes e das tendências de pesquisa sobre o gerenciamento de contêineres. Os autores analisam e comparam técnicas de escalonamento de contêineres em quatro categorias:

- **Modelagem matemática:** Técnicas que utilizam modelos matemáticos para representar o problema de escalonamento e alcançar a solução ótima;
- **Heurísticas:** Segundo os autores, a maioria das técnicas relatadas utiliza heurísticas para encontrar soluções aproximadas para o problema. São técnicas que usam regras simples para gerar soluções sub-ótimas em menor tempo;
- **Meta-heurísticas:** Trata-se de extensões das heurísticas que usam estratégias de busca mais sofisticadas para encontrar soluções melhores;

- **Aprendizagem de máquina:** São técnicas que utilizam algoritmos capazes de aprender com dados históricos e gerar soluções mais eficientes.

Ao analisar diferentes abordagens em concordância com as categorias apresentadas para diferentes objetivos, identificando seus benefícios, limitações e desafios, os autores concluem que não existe uma técnica ideal para todas as situações, e que a escolha da técnica mais adequada depende de fatores como o tipo de carga de trabalho, os recursos de hardware disponíveis e os objetivos de desempenho e custo.

[Rodrigues et al. 2019] apresenta o EMULAG, um escalonador centralizado que tem como objetivo principal a consolidação de contêineres em um DC. A implementação faz uso do método de Processo Hierárquico Analítico (AHP) [Shim 1989] com o algoritmo *Markov Cluster Algorithm* (MCL). Desenvolvido na década de 1970, o AHP é um método utilizado para tomada de decisão que possibilita escolher a melhor opção entre várias alternativas com base em múltiplos critérios. O MCL é um algoritmo heurístico de *clustering* usado para detectar comunidades em grafos, baseado em fluxo de dados e Cadeia de Markov [Van Dongen 2000]. O EMULAG faz uso de ambos os algoritmos, AHP e MCL, processados em unidades de processamento gráfico (GPU). A abordagem apresentada pelos autores realiza o agrupamento de recursos do DC com MCL, utilizando o AHP para classificar os grupos construídos e selecionar os servidores conforme os resultados obtidos na classificação.

Ao analisar o estudo apresentado pelos autores, é possível concluir que EMULAG é sensível quanto à configuração adequada de seus parâmetros, como pesos para critérios no AHP. O ajuste incorreto dos parâmetros impacta consideravelmente os resultados, o que pode considerar esse como um dos pontos de maior atenção na implementação. Além disso, ao analisar os resultados obtidos pelos autores, é possível observar que os atrasos obtidos pelo escalonador podem ser um fator de desvantagem considerável para um DC.

[Li et al. 2018] apresenta uma abordagem de escalonamento de contêineres baseada em PSO para a plataforma Docker Swarm [Docker 2023]. Com o objetivo de evitar os problemas de baixa utilização de recursos e distribuição inadequada de carga no orquestrador, a abordagem possui um ponto bastante interessante. Os autores destacam a possibilidade de o PSO ficar preso em uma solução ótima local durante o processo de busca. Como medida para sanar o problema, os autores combinaram o PSO à técnica conhecida como *Simulated Annealing*, uma meta-heurística inspirada no processo físico de recozimento de metais [Van Laarhoven et al. 1987].

Os experimentos avaliam a implementação comparando-a algoritmos *Spread* e *Random* do próprio Docker Swarm. Os resultados obtidos para o balanceamento de carga foram 20% melhores quando comparados ao *Spread* e 19% quando comparados ao *Random*, porém o trabalho se limita aos problemas do Docker Swarm, sendo interessante realizar testes para escalonamentos mais abrangentes.

[Ben Alla et al. 2017] apresenta o DPQ-PSO, um algoritmo que escalona contêineres utilizando fila de prioridade dinâmica com foco no balanceamento de carga e na utilização dos recursos. A abordagem tem como base principal o AHP em conjunto com o PSO. A prioridade de uma tarefa é medida com base em um modelo de tomada de decisão de critérios múltiplos usando o AHP para atribuir prioridades dinâmicas às tarefas. Além disso, o método dos quartis, uma medida estatística que divide um

conjunto de dados em quatro partes iguais, cada uma representando 25% dos dados [Bussab and Morettin 2010], é aplicado para classificar as tarefas em diferentes níveis de prioridade (Baixa, Média e Alta) com base na sua urgência e importância relativas. Essas prioridades dinâmicas são então utilizadas no processo de agendamento para determinar a ordem em que as tarefas são despachadas e executadas, visando otimizar o desempenho do sistema. Já o PSO é utilizado para encontrar uma solução de escalonamento. As tarefas requisitadas são categorizadas pelo AHP, como mencionado, e submetidas ao processo de escalonamento, processo esse que tem como base principal o PSO. Embora classificar as tarefas por prioridade seja uma abordagem bastante interessante, um ponto importante deve ser observado. Por ser um algoritmo iterativo, o tempo de classificação das tarefas para então dividi-las em filas pode afetar o tempo de resposta do algoritmo, considerando o tempo necessário para o AHP construir as filas, principalmente por conta dos cálculos para obter o índice de consistência durante essa construção. Outro ponto importante a se destacar são os experimentos realizados. Além disso, os autores consideraram, em seus experimentos, conjuntos de tarefas de tamanhos 5, 10, 15 e 25, o que abre espeço para experimentos mais robustos.

Durante o levantamento de trabalhos na literatura, não foram encontradas implementações que combinem o uso do algoritmo PSO juntamente com o *Path Relinking* aplicadas ao problema de alocação abordado no presente trabalho. Adicionalmente, vale ressaltar que o segundo estudo apresentado é o único que se dedica à consolidação de contêineres em DC. Desta forma, a próxima seção deste trabalho se dedica a fornecer uma análise mais aprofundada sobre a implementação em si. São apresentados os detalhes técnicos e práticos do algoritmo proposto, destacando como os aprendizados adquiridos com os estudos correlatos influenciaram as escolhas de implementação.

4. Abordagem proposta baseada em PSO com *Path Relinking*

A tarefa de alocar contêineres para consolidar um DC apresenta semelhanças significativas com o conhecido Problema das Múltiplas Mochilas (MKP). Ambos envolvem a alocação eficiente de itens em um espaço limitado. No MKP, o conjunto de itens disponíveis pode ser facilmente representado como um vetor de binários em que cada elemento do vetor pode assumir o valor 0 ou 1, sendo o valor 0 para itens que não fazem parte da solução, e 1 para itens que fazem parte da solução. A função objetivo definida para o contexto do presente estudo visa minimizar a proporção de *nós* ativos que recebem pelo menos um contêiner em relação ao total de *nós* no DC. Essa proporção é expressa como $\frac{|S_a|}{|S|}$, onde S_a representa os *nós* ativos e S denota o conjunto de todos os *nós* no DC.

Para alcançar o objetivo estabelecido, a abordagem desenvolvida se baseia na meta-heurística PSO, em conjunto com a estratégia de intensificação da busca *Path Relinking*. A implementação tem sua abordagem principal fundamentada nas iterações do PSO. A velocidade e a posição de cada partícula têm como base as seguintes equações:

$$v_{ij}^{k+1} = \omega v_{ij} + c_1 r_1 (pbest_{ij}^k - x_{ij}^k) + c_2 r_2 (gbest_{ij}^k - x_{ij}^k) \tag{1}$$

$$x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1} \tag{2}$$

Nas Equações 1 e 2, os valores de i e j referem-se, respectivamente, ao identificador da partícula e sua dimensão. Na Equação 1, ω é o coeficiente de inércia, utilizado para controlar a influência de resultados de velocidade já calculados. Ainda na Equação 1, o valor x_{ij}^{k} representa a posição da partícula i até o momento. Já v_{ij}^{k+1} representa a velocidade da partícula i na iteração $(k+1)$.

Os valores de $pbest_{ij}^{k}$ e $gbest_{ij}^{k}$, correspondem, na devida ordem, ao melhor resultado da função *fitness* para a posição da partícula i até o momento e o melhor resultado entre todas as partículas do exame até o momento. Na Equação 2, x_{ij}^{k+1} refere-se à posição da partícula i na iteração $(k+1)$.

4.1. Implementação do PSO

A abordagem proposta no presente estudo utiliza como base a implementação do trabalho de Li *et al.*(2018). A representação das partículas da população estruturada pelos autores é definida por uma codificação de números naturais. Com isso, supondo que seja necessário alocar m contêineres em n nós, uma partícula i é codificada por $P = \{p_1, p_2, ..., p_m\}$. Para ilustrar essa implementação, a Figura 14 exemplifica uma partícula para 8 contêineres $(m = 8)$ e 5 *nós* $(n = 5)$. Cada partícula é representada por um *array* de números inteiros no qual os índices representam os contêineres a serem alocados e o valor de cada índice representa o *nó* onde o contêiner é alocado.

Contêineres	1	2	3	4	5	6	7	8
Nós	3	2	5	4	3	1	2	2

Figura 1. Codificação das partículas [Li et al. 2018].

Definida a estrutura das partículas, a atualização da posição de cada uma é um ponto fundamental para o algoritmo PSO. O Algoritmo 1 ilustra o ponto da implementação desenvolvida, responsável pela atualização da posição de cada uma das partículas. Seguindo as Equações 1 e 2, as variáveis *inertia*, $c1$, e $c2$ representam os coeficientes que controlam a influência da inércia, da influência cognitiva e da influência social, respectivamente. As variáveis $r1$ e $r2$ são números aleatórios no intervalo $(0, 1]$ responsáveis por adicionar aleatoriedade no movimento das partículas.

Algoritmo 1: Atualização da velocidade e posição das partículas

```
1 para cada particula faça
2     para i de 0 até a quantidade de contêineres faça
3         partícula.velocidade[i] = (inercia * partícula.velocidade[i] + c1 * r1 *
            (partícula.posição[i] + c2 * r2 * (gbest[i] - partícula.posição[i]))
4         partícula.posição[i] = max(0, min(quantidadeDeServidores, partícula.posição[i] +
            partícula.velocidade[i]))
```

No PSO, cada partícula tem sua posição ajustada em busca da solução ótima. Esse ajuste tem base em uma função de avaliação específica, comumente conhecida como função *fitness*. A implementação da função *fitness* considerada neste trabalho é apresentada no Algoritmo 2.

A implementação considera a capacidade de cada recurso disponível em cada *nó* como a quantidade restante do recurso mais alocado. Por exemplo, supondo que a CPU seja o recurso com menor capacidade disponível em um dado *nó*, a capacidade de alocação considerada é a quantidade que falta para atingir sua capacidade máxima de CPU. É importante destacar que a implementação desenvolvida considera em suas iterações os recursos de CPU, memória RAM, armazenamento e largura de banda. Isso se deve ao impacto significativo que eles exercem no desempenho das aplicações hospedadas pelos contêineres.

Algoritmo 2: Função Fitness do PSO

Entrada: partícula

1 $listaDeContainers \leftarrow$ partícula.listaDeContêineres;
2 $listaDeVMs \leftarrow$ partícula.listaDeVMs;
3 $posicaoParticula \leftarrow$ partícula.posicao;
4 **início**
5 $novasCapacidadesVMs \leftarrow$ Lista vazia de pesos de servidor;
6 **para** i *de* 0 *até o tamanho de* $listaDeVMs$ **faça**
7 Adicionar 0.0 a $novasCapacidadesVM$;
8 **para** i *de* 0 *até o tamanho do array* $posicaoParticula$ **faça**
9 $item \leftarrow posicaoParticula[i]$;
10 $novasCapacidadesVMs[item] \leftarrow novasCapacidadesVMs[item] +$ calculaConsumo$(containerList[i], containerVmList[item])$;
11 $totalPenalty \leftarrow 0.0$;
12 **para** i *de* 0 *até o tamanho de* $novasCapacidadesVMs$ **faça**
13 $weight \leftarrow novasCapacidadesVMs[i]$;
14 $totalPenalty \leftarrow totalPenalty + \max(0.0, weight - 100.0)$;
 Saída: $totalPenalty$;

Para garantir a eficiência operacional requerida por cada contêiner, é considerada a quantidade máxima estabelecida para cada um desses recursos. Inicialmente, todos os valores da variável do tipo lista $novasCapacidadesVMs$, que representa as capacidades disponíveis em cada servidor na partícula atual, são definidos como zero. Em seguida, iterando sobre as posições da partícula(Linhas 8-10), onde o consumo de um contêiner em um *nó* específico é calculado pelo método *calculaConsumo* e atualiza o peso correspondente na lista $novasCapacidadesVMs$. A função verifica a que *nós* o item pertence e adiciona o peso desse item ao peso total do *nós* correspondente na lista $novasCapacidadesVMs$.

Caso um *nó* tenha sua capacidade excedida, este contribui para a penalização, a qual é a soma das diferenças entre o peso total e a capacidade de cada servidor. No entanto, se um servidor estiver dentro da capacidade, estabelecida como 100%, a diferença é definida como zero na penalização (Linhas 11-14). O valor da solução que a partícula representa é calculado subtraindo a penalidade do valor total. Levando em conta que esta função desempenha o papel de avaliar todas as soluções dentro da abordagem desenvolvida, ela é igualmente empregada pelo *Path Relinking*.

4.2. *Path Relinking*

A implementação do *Path Relinking* realizada no presente trabalho é embasada no algoritmo proposto por Menezes *et al.* (2017). Na Figura 2, o ponto (A) ilustra a primeira iteração do *Path Relinking*. A primeira parte da solução corrente, que no primeiro momento é a inicial, é substituída pela primeira parte da solução guia, gerando a primeira do trajeto. Em (A), (B) e (C) a função objetivo sempre é aplicada para a solução corrente e o melhor resultado é atualizado, caso necessário. Já (D) denota o ponto de parada, quando a solução atual coincide com a solução guia.

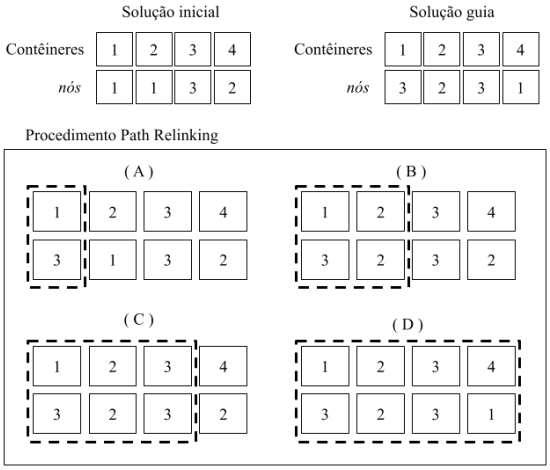

Figura 2. Procedimento Path Relinking em conjunto com o PSO.

4.3. Processo de alocação

Tendo como base o estudo de Alla *et al.* (2017), a abordagem desenvolvida no presente estudo realiza a alocação dividindo em blocos os contêineres durante o processo de alocação. Isso é possível porque as partículas do PSO representam potenciais alocações, o que resulta em um número maior de contêineres alocados.

A alocação de contêineres por blocos de requisições aproveita as vantagens que o PSO possui quanto a exploração do espaço de buscas. Uma vez que a quantidade de configurações possíveis torna-se proporcional, além da quantidade de partículas, ao tamanho do bloco que se deseja alocar. Com isso, a possibilidade de atender uma quantidade maior de requisições acaba sendo maior. Na implementação realizada para o presente trabalho, uma requisição de alocação é dividida em blocos antes de determinar qual *nó* comportará cada contêiner da requisição. O tamanho de cada bloco é definido previamente e cada um é tratado como uma unidade independente.

5. Experimentos

Para avaliar a política proposta, foram conduzidos experimentos utilizando o simulador CloudSim [Calheiros et al. 2011]. O simulador suporta tanto a modelagem quanto a simulação para nuvens individuais e interconectadas, permitindo modelagem contínua, experimentos e simulação de sistemas de computação em nuvem e ambientes de provisionamento de aplicativos. No enatando, é importante mencionar que para experimentos

utilizando contêineres o CloudSim utiliza VMs para alocar os contêineres, sendo uma limitação deste.

As simulações realizadas foram definidas com base no trabalho de Alla et al.(2017). Foram consideradas alocações de tamanho variável, especificamente contendo 10, 30, 50, 80 e 100 contêineres em DCs com 10 *hosts*, todos com largura de banda de 1 Gbit/s, e 50 máquinas virtuais, todas com largura de banda de 10Mbits/s, e demais especificações apresentadas nas Tabelas 1 e 2. Como dados de carga de trabalho, foram utilizados todos os dados do PlanetLab disponibilizados pelo próprio CloudSim. É válido destacar que os dados foram utilizados de forma aleatória com o intuito de explorar a avaliação da política com maior robustez.

Tabela 1. Especificações das VMs utilizadas nos experimentos.

Número de VMs	MIPS do processador	Memória RAM(MB)
13	18.637	1.024
13	18.637	2.048
13	18.637	4.096
11	18.637	8.192

Tabela 2. Especificações dos *hosts* utilizadas nos experimentos.

Número de *hosts*	MIPS do processador	Memória RAM(MB)
4	149.096	65.536
4	298.192	131.072
2	596.384	262.144

5.1. Métricas

Conforme o objetivo da política de alocação desenvolvida neste estudo, que tem como foco a consolidação do DC, foram consideradas as seguintes métricas:

- **Consumo energético:** A escolha desta métrica é amparada pela afirmação contida no estudo de Helali e Omri (2021) de que a redução dos ativos de TI impacta na redução de custos energéticos. A métrica é obtida pela coleta de dados sobre o consumo de energia de servidores que fazem parte da infraestrutura da nuvem.
- **Quantidade de VMs utilizadas após o processo de alocação:** Uma vez que uma política ideal deve alocar recursos de forma eficiente, evitando a superutilização de VMs é algo fundamental, haja visto que um número excessivo de VMs pode levar a custos desnecessários de recursos CPU e memória, além de aumentar o consumo de energia. Desta forma, a quantidade de VMs utilizadas torna-se uma métrica importante para avaliar a eficiência da política de alocação.
- **Média de violações de Acordo de Nível de Serviço (SLA) nas VMs:** A escolha desta métrica tem como base o trabalho de e Mann e Szabó (2017). O Acordo de Nível de Serviço, em inglês *Service Level Agreements* (SLA), trata-se de um contrato que estabelece um nível de expectativas, por parte do cliente, dos serviços assumido por provedor. Uma vez que nas simulações realizadas, os contêineres são alocados em máquinas virtuais, a média de violações de SLA nas VMs selecionadas é uma métrica de suma importância para avaliar o quão bem a política está mantendo os níveis de serviço acordados.

A política desenvolvida é comparada com políticas que têm como base os algoritmos FCFS *(First-Come, First-Served)*, *random* e PSO. As configurações de número de partículas, quantidade de iterações e coeficiente de inércia são aplicadas tanto na política PSOPR, quanto na política implementada utilizando apenas o PSO. Por meio de experimentos utilizando busca aleatória, os parâmetros do PSO foram definidos em número de partículas estabelecido em 80, 10 iterações, blocos comportando até 3 contêineres e inércia de valor 0.8. Além disso, cada tamanho de alocação foi submetido a 100 testes independentes.

5.2. Resultados

A Tabela 3 apresenta os resultados de média e desvio padrão dos experimentos para as métricas de consumo energético, violação de SLA e tempo de resposta para cada política de alocação considerada nos experimentos. Sobre a métrica de consumo energético, conforme os dados apresentados, a política desenvolvida neste estudo, identificada como PSOPR, obteve a melhor média de consumo de energia. Ao analisar as médias apresentadas na tabela, é perceptível que, embora os valores das políticas PSOPR e PSO estejam abaixo das médias das demais políticas avaliadas, há uma tendência de alocações mais elevadas para 10 contêineres. A média do consumo energético é maior após a alocação de 10 contêineres do que a alocação de 50 contêineres, como se pode observar na Tabela 3. Ao analisar as VMs utilizadas nesses experimentos após as alocações, observou-se que as VMs de maior capacidade foram selecionadas. Essa alocação de VMs mais robustas pode resultar em um consumo energético mais eficiente. No entanto, para uma compreensão mais aprofundada dos resultados, é necessário realizar uma investigação mais detalhada.

Tabela 3. Resultados experimentais de comparação das políticas de alocação em termos de consumo energético, violação de SLA e tempo de resposta.

Políticas de Alocação	Número de Contêineres	Energia Consumida (kW)		Violação de SLA (%)		Tempo de Resposta (segs)	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
PSOPR	10	19,23	3,16	2%	4%	3,47	0,46
	30	15,18	4,3	9%	2%	12,02	31,69
	50	17,45	3,94	9%	2%	16,31	12,02
	80	19,13	4,26	9%	2%	27,74	2,86
	100	24,82	4,24	8%	4%	36,54	4,95
PSO	10	19,62	3,5	3%	4%	4,39	1,49
	30	17,05	4,28	9%	2%	7,07	0,83
	50	18,25	3,83	10%	1%	9,09	1,08
	80	20,21	4,39	10%	2%	13,94	1,56
	100	26,46	4,57	9%	3%	15,47	1,58
FCFS	10	25,85	0,06	0%	0%	0,27	0,16
	30	25,10	0,7	9%	3%	0,34	0,24
	50	25,68	0,4	9%	3%	0,35	0,22
	80	25,82	0,03	10%	0%	0,62	0,78
	100	25,82	0,36	10%	1%	0,73	1,08
random	10	24,84	3,45	4%	5%	2,21	1,92
	30	25,36	2,08	10%	3%	3,33	2,14
	50	25,66	1,4	10%	1%	3,70	1,98
	80	25,74	0,72	10%	1%	4,36	2,25
	100	25,52	1,42	10%	1%	0,56	0,24

Conforme os dados apresentados na Tabela 3, acerca dos resultados de violação de SLA, no geral, todas as políticas de alocação apresentaram, em média, valores de violação máxima em torno de 10%. Isso significa que, em média, o serviço dos contêineres alocados está disponível na maioria das vezes, com apenas uma pequena porcentagem de casos

em que ocorre uma violação do SLA. Contudo, é importante ressaltar que a definição de uma taxa de violação de SLA aceitável pode variar conforme o serviço ou sistema em questão. O Amazon S3, por exemplo, fornece compromisso de serviço aos usuários, garantindo $99,9\%$ de tempo de atividade por mês [Zhou et al. 2019]. Nos experimentos realizados, a política FCFS obteve desempenho superior em termos de violação do SLA quando uma quantidade reduzida de contêineres foi alocada. No entanto, à medida que o número de contêineres aumenta, a política desenvolvida neste estudo se mostrou mais eficaz. Além disso, os dados apresentados mostram valores constantes obtidos nos experimentos em que a política *random* foi aplicada. Esses resultados implicam que mais mecanismos inerentes ao gerenciamento de recursos do DC, como, por exemplo, as políticas de alocação e seleção de VMs, contribuíram para a obtenção desses resultados constantes.

Como mencionado anteriormente, a quantidade de máquinas virtuais alocadas é uma métrica fundamental para avaliar a eficiência de uma política de alocação de contêineres. A Figura 3 fornece uma representação visual da quantidade média de VMs alocadas nos experimentos realizados. Ao analisar o gráfico, é perceptível que as políticas que possuem o PSO como base demonstraram resultados superiores em comparação com as demais. Os resultados indicam que a política de alocação desenvolvida neste trabalho foi capaz de otimizar o uso das máquinas virtuais e evitar que o impacto de ótimos locais para o PSO, resultando em uma alocação mais eficiente dos contêineres.

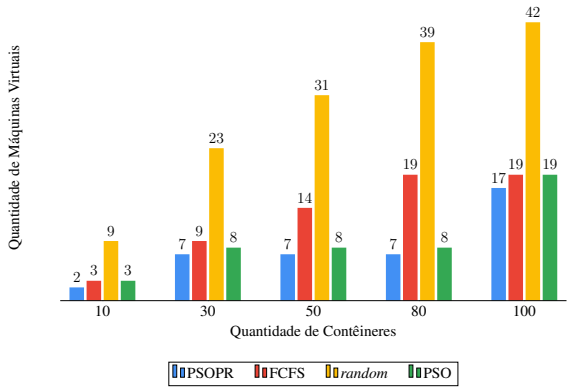

Figura 3. Médias da quantidade de VMs utilizadas após o processo de alocação.

Embora não seja uma métrica priorizada neste trabalho, a eficiência de tempo, ou seja, o tempo que um algoritmo necessita para processar uma entrada e produzir uma saída, tem potencial para disponibilizar informações valiosas sobre implementações. Em alguns casos, implementações que demonstram excelente desempenho em outras métricas podem ser comprometidas quando se trata do elemento temporal, sendo o caso das implementações PSOPR e PSO implementadas neste trabalho. O tempo necessário para alocar os contêineres requisitados, apresentados na Tabela 3, é muito superior ao tempo que as demais políticas submetidas aos testes necessitam. É importante deixar claro que, embora as políticas PSOPR e PSO necessitem de mais tempo, elas produzem decisões melhores, como pode ser observado na Tabela 3 e na Figura 3.

6. Considerações Finais

Este trabalho apresenta uma abordagem desenvolvida com base na combinação do algoritmo *Particle Swarm Optimization* (PSO) em conjunto com a meta-heurística *Path*

Relinking. Essa combinação é aplicada ao problema de gerenciamento de recursos em ambientes de computação em nuvem. A complexidade do problema abordado no presente estudo é evidenciada no estudo de Ahmad *et al*. (2021), enfatizando ainda mais a importância de implementações robustas aplicadas ao problema..

Por meio dos experimentos realizados, foi possível mostrar que a política de alocação desenvolvida neste estudo se destacou frente às métricas consideradas nos experimentos realizados. Os resultados obtidos foram superiores em comparação com as abordagens convencionais para as métricas de consumo de energia elétrica, quantidade de máquinas virtuais utilizadas para alocar uma determinada quantidade de contêineres e a violação do SLA. Os resultados apresentados mostram ainda que a utilização exclusiva do PSO no contexto da alocação de contêineres no cenário de computação em nuvem não é tão interessante quanto sua combinação deste com outras abordagens. A melhora na precisão dos resultados ao utilizar o PSO em conjunto com o *Path Relinking* deve-se, na maioria, à implementação que conseguiu evitar ótimos locais durante os processos que envolvem funcionamento do PSO.

Os resultados do desvio padrão obtidos a partir dos testes realizados evidenciam a sensibilidade dos algoritmos que possuem o PSO como base. Essa sensibilidade, em parte, é certamente um dos principais fatores que influenciaram no tempo de execução. A quantidade de parâmetros que necessitam de ajuste e a complexidade das restrições envolvidas impactam diretamente no desempenho ao explorar um grande espaço de busca. No entanto, esses resultados abrem caminho para uma implementação mais eficaz, tendo como base essa meta-heurística. Desta forma, como trabalhos futuros, além de investigar a utilização do PSO em conjunto com outras abordagens heurísticas ou a viabilidade de uma implementação em conjunto com técnicas de aprendizagem de máquina para ajustar os parâmetros que fazem parte do funcionamento do PSO, bem como os tamanhos dos blocos utilizados pela implementação desenvolvida neste estudo. Adicionalmente, também é válido realizar análises mais profundas sobre a escalabilidade da solução proposta e sua aplicabilidade em diferentes cenários de nuvem, realizando experimentos em ambientes diferentes do CloudSim, que aloca contêineres em VMs.

Referências

Ahmad, I., AlFailakawi, M. G., AlMutawa, A., and Alsalman, L. (2021). Container scheduling techniques: A survey and assessment. *Journal of King Saud University-Computer and Information Sciences*.

Ben Alla, H., Ben Alla, S., Ezzati, A., and Touhafi, A. (2017). An efficient dynamic priority-queue algorithm based on ahp and pso for task scheduling in cloud computing. In Abraham, A., Haqiq, A., Alimi, A. M., Mezzour, G., Rokbani, N., and Muda, A. K., editors, *Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016)*, pages 134–143, Cham. Springer International Publishing.

Bussab, W. d. O. and Morettin, P. A. (2010). Estatística básica. In *Estatística básica*, pages xvi–540.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

de Assuncao, M. D., da Silva Veith, A., and Buyya, R. (2018). Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *Journal of Network and Computer Applications*, 103:1–17.

Docker (2023). Enterprise container platform for high-velocity innovation.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29(3):653–684.

Helali, L. and Omri, M. N. (2021). A survey of data center consolidation in cloud computing systems. *Computer Science Review*, 39:100366.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Li, L., Chen, J., and Yan, W. (2018). A particle swarm optimization-based container scheduling algorithm of docker platform. In *Proceedings of the 4th International Conference on Communication and Information Processing*, pages 12–17.

Mann, Z. A. and Szabó, M. (2017). Which is the best algorithm for virtual machine placement optimization? *Concurrency and Computation: Practice and Experience*, 29(10):e4083.

Menezes, M. d. S., Goldbarg, M. C., Goldbarg, E. F. G., Ferreira, V. E. S., and Correia, G. C. (2017). Abordagens grasp aplicadas ao problema quota cars. In *Anais do 49 Simpósio Brasileiro de Pesquisa Operacional*, pages 1807–1818.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Rodrigues, L., Pasin, M., Alves Jr, O. C., Pillon, M. A., Miers, C. C., and Koslovski, G. P. (2019). Escalonamento de contêineres com método de decisao multicritério acelerado por gpu. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 515–528. SBC.

Shim, J. P. (1989). Bibliographical research on the analytic hierarchy process (ahp). *Socio-Economic Planning Sciences*, 23(3):161–167.

Van Dongen, S. (2000). Graph clustering by flow simulation. *PhD thesis, University of Utrecht*.

Van Laarhoven, P. J., Aarts, E. H., van Laarhoven, P. J., and Aarts, E. H. (1987). *Simulated annealing*. Springer.

Varghese, B. and Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

Zhou, X., Li, K., Liu, C., and Li, K. (2019). An experience-based scheme for energy-sla balance in cloud data centers. *IEEE Access*, 7:23500–23513.