Thompson Sampling in Heuristic Selection for the Quadratic Assignment Problem

Carlos Daniel Pohlod[1], **Sandra M. Venske**[1], **Carolina P. Almeida**[1]

[1]Departamento de Ciência da Computação (DECOMP)
Universidade Estadual do Centro-Oeste (UNICENTRO)
Guarapuava – PR – Brasil

carlospohlod@gmail.com, {carol, ssvenske}@unicentro.br

Abstract. *In this work we propose a selection Hyper-Heuristic (HH) based on the Thompson Sampling (TS) approach to the solution of Quadratic Assignment Problem (QAP). The QAP aims to allocate facilities in a set of possible known locations, in order to minimize the total cost of all movements between facilities. The proposed HH is applied in the automatic configuration of a memetic algorithm, acting on the selection of a combination of low-level heuristics. Each combination involves the selection of a recombination heuristic, a local search strategy and a mutation heuristic. The algorithm was tested in 15 instances of Nug benchmark and the performance of HH is superior to that obtained by any combination of heuristics applied in isolation, demonstrating its efficiency in the automatic configuration of the algorithm. Experiments show that TS performance is affected by the quality of the set of low-level heuristics. The best version of HH obtains the optimal solution in 9 instances and the mean percentage deviation of the optimal solution (gap), considering all 15 instances, was 8.6%, with the largest gaps found for the three largest instances.*

Resumo. *Este trabalho propõe uma Hiper-Heurística (HH) de seleção baseada na abordagem Thompson Sampling (TS) para a solução do Problema Quadrático de Alocação (PQA). O PQA tem como objetivo a alocação de instalações em um conjunto de possíveis localidades já conhecidas, a fim de minimizar o custo total de todas as movimentações entre as instalações. A HH proposta é aplicada na configuração automática de um algoritmo memético, atuando na seleção de uma combinação de heurísticas de baixo nível. Cada combinação envolve a seleção de uma heurística de recombinação, de uma estratégia de busca local e de uma heurística de mutação. O algoritmo foi analisado em 15 instâncias do benchmark Nug e o desempenho da HH é superior àquele obtido por qualquer combinação de heurísticas aplicada de forma isolada, demonstrando a sua eficiência na configuração automática do algoritmo. Os experimentos mostram que o desempenho da TS é afetado pela qualidade do conjunto de heurísticas de baixo nível. A melhor versão da HH obtém a solução ótima em 9 instâncias e o desvio médio percentual da solução ótima (gap), considerando todas as 15 instâncias foi de 8,6%, sendo que os maiores gaps foram encontrados para as três maiores instâncias.*

1. Introdução

A Computação Evolucionária (CE) faz parte dos diversos subcampos da inteligência artificial, inspirando-se na evolução natural das espécies bem como nas teorias Darwi-

nistas e Lamarckianas [Engelbrecht 2007, Moreira 2011]. A CE traz a implementação de heurísticas/algoritmos computacionais capazes de encontrar soluções locais e/ou globais satisfatórias [de Oliveira Neto 2017]. O desempenho dos diversos algoritmos evolucionários abordados na literatura é dependente do problema abordado e pode variar de acordo com fatores como parâmetros ou instâncias do problema. Portanto, trata-se de um desafio ao pesquisador encontrar as melhores configurações e combinações destes algoritmos de forma a atingir um resultado satisfatório. Abordando esta questão, as Hiper-heurísticas (HH) [Burke et al. 2013, Drake et al. 2020] são técnicas que trabalham indiretamente com a função objetivo, construindo (hiper-heurística de geração) ou selecionado (hiper-heurística de seleção) heurísticas para percorrer o espaço de busca de um problema. Em uma HH de seleção, a qual é considerada neste trabalho, ocorre a escolha automática de heurísticas de baixo nível, tais como, por exemplo, operadores de reprodução e de mutação, busca local e outros parâmetros [Sucupira 2007] por meio de uma heurística de alto nível. A heurística de alto nível pode atuar de diferentes maneiras, podendo ser aleatória [Rodrigues 2017] ou utilizando métodos inteligentes de escolha, como raciocínio baseado em casos, busca local e *Multi-Armed Bandit* (MAB) [Sucupira 2007, Pillay and Qu 2018].

As HHs têm sido aplicadas com sucesso em diferentes problemas de otimização combinatória [Pillay and Qu 2018]. Nesse trabalho, o problema de otimização tratado é o Problema Quadrático de alocação (PQA). Proposto por [Beckman and Koopmans 1957], o PQA é um problema combinatório da classe NP-difícil capaz de modelar diversos problemas do mundo real [Çela 2013], despertando interesse de diversos autores. Dentre as possíveis aplicações do PQA estão problemas de fiação em eletrônica, computação paralela e distribuída, análise estatística de dados, *design* de painéis de controle e teclados para máquinas de escrever, esportes, química, arqueologia e balanceamento de turbinas, fabricação de computadores e transporte [Çela 2013, Loiola et al. 2004]. Por ser um problema da classe NP-difícil, o PQA é computacionalmente inviável, principalmente para instâncias médias e grandes [de Oliveira Neto 2017], sendo então um foco de estudo da CE.

Com o intuito de demonstrar a efetividade das HHs na escolha de heurísticas, o presente trabalho traz a implementação uma HH baseada na abordagem *Thompson Sampling* para a solução do PQA utilizando as instâncias Nug [Nugent et al. 1968]. A HH implementada opera sobre um Algoritmo Memético (AM), ou seja, um algoritmo genético que faz uso de buscas em espaço local, além do global. O papel da HH é escolher uma combinação composta por heurísticas de recombinação, busca local e mutação, a ser aplicada na geração de soluções para a próxima geração do AM. As principais contribuições deste trabalho envolvem: i) A criação de uma HH capaz de selecionar heurísticas de diferentes categorias em um AM; ii) Análise do desempenho da HH proposta no problema quadrático de alocação; e iii) Verificação do impacto da alteração na qualidade do conjunto de heurísticas a serem selecionadas pela *Thompson Sampling* em seu desempenho. Os experimentos mostram que a HH supera as combinações de heurísticas aplicadas de forma isolada, tendo o seu desempenho afetado pela qualidade do conjunto de heurísticas de baixo nível. A melhor versão da HH obtém a solução ótima em 9 instâncias e o desvio médio percentual da solução ótima (*gap*), considerando todas as 15 instâncias foi de 8,6%, sendo que os maiores *gaps* foram encontrados para as três maiores instâncias.

O restante deste artigo está organizado como segue. A Seção 2 introduz os fundamentos básicos relacionados ao problema abordado (PQA) e às HHs. Na Seção 3 é apresentada a metodologia proposta para o desenvolvimento da HH para o PQA. Os resultados são mostrados e discutidos na Seção 4. Finalmente, a Seção 5 encerra o trabalho com as conclusões e futuros direcionamentos.

2. Fundamentação teórica

Nesta seção são apresentados brevemente o Problema Quadrático de Alocação e os conceitos e abordagens relacionados às Hiper-Heurísticas.

2.1. Problema Quadrático de Alocação (PQA)

O PQA foi introduzido por [Beckman and Koopmans 1957] e é um exemplar da classe problemas NP-difícil [Sahni and Gonzales 1976]. O PQA é um dos problema mais difíceis da área de otimização combinatória, pois instâncias de tamanho maior ou igual a 20 não podem ser resolvidas de maneira ótima em tempo computacional aceitável, o que justifica o uso de métodos aproximativos para o tratamento deste problema [Knowles and Corne 2003].

O PQA considera um conjunto de n instalações e um conjunto de n localidades. Para cada par de localidades é especificada uma distância, enquanto para cada par de instalações é especificado um valor de fluxo. O fluxo representa a quantidade de suprimentos transportados entre as duas instalações. O problema corresponde em associar todas as instalações a diferentes localidades com o objetivo de minimizar a soma das distâncias multiplicadas pelo fluxo correspondente [Sahni and Gonzales 1976]. Considerando uma matriz de distâncias D = $\{d_{ij}\}_{n\times n}$ e uma matriz de fluxos B = $\{b_{ij}\}_{n\times n}$, o PQA pode ser definido formalmente da seguinte maneira [Knowles and Corne 2003]:

$$\underset{\pi}{Minimizar}\ C(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} b_{\pi_i\pi_j} \tag{1}$$

tal que n é o número de instalações/localidades, d_{ij} é a distância entre a localidade π_i e a localidade π_j, $b_{\pi_i\pi_j}$ é o fluxo da facilidade i para a facilidade j e π_i representa a facilidade na localidade i na permutação $\pi \in P(n)$, sendo $P(n)$ o conjunto de todas as permutações $\{1, 2, \ldots, n\}$.

2.2. Hiper-Heurísticas (HH)

As Hiper-heurísticas são técnicas recentes que visam encontrar soluções adequadas para diversos problemas de otimização do mundo real [Pillay and Qu 2018]. As HHs trabalham no espaço de busca das heurísticas ao invés do espaço de soluções, selecionando (HH de seleção) ou gerando (HH de geração) heurísticas de baixo nível que são usadas para resolver o problema em questão [Burke et al. 2013].

Uma HH é composta por uma estratégia de alto nível, a qual utiliza um mecanismo para decidir qual heurística de baixo nível utilizar, e um critério de aceitação que pode aceitar ou rejeitar uma solução gerada. O outro componente da HH é o conjunto de heurísticas de baixo nível, que devem ser apropriadas para o problema testado [Sabar et al. 2015]. As heurísticas de baixo nível podem ser de dois tipos, construtivas ou perturbativas. As heurísticas construtivas são usadas para criar uma solução para um

problema. Heurísticas perturbativas são aplicadas para aprimorar uma solução existente. As HHs podem aprender *online*, quando o aprendizado se dá durante a execução do algoritmo, ou *offline* quando o aprendizado já vem com as regras definidas antes da execução do algoritmo. A utilização das HHs se dá por conta de sua capacidade de encontrar bons resultados para problemas NP-difíceis, retirando do usuário a necessidade de escolher uma única heurística ou a sua configuração [Drake et al. 2020].

Neste trabalho a HH proposta aprende *online* e considera a abordagem *Thompson Sampling* como heurística de alto nível. No baixo nível são usadas diferentes combinações de heurísticas perturbativas de recombinação, busca local e mutação. O critério de aceitação permite que somente modificações que trazem melhorias sejam aceitas.

2.2.1. Heurística de Alto Nível - *Thompson Sampling* (TS)

Thompson Sampling [Thompson 1933], um exemplar do método *Multi-Armed Bandit*, é uma abordagem utilizada para a tomada de decisão que funciona a partir da coleta de informações sobre as ações realizadas, de forma que haja um equilíbrio entre a exploração do que é conhecido e o investimento no acúmulo de novas informações, com o objetivo de melhorar o desempenho futuro do processo de busca. A abordagem *Thompson Sampling* pode utilizar um modelo Bayesiano para representar a incerteza associada a uma tomada de decisão sobre K ações (combinações de heurísticas) [Russo et al. 2021]. Toda a ação $A = k$ produz, no contexto β-Bernoulli, uma recompensa igual a 1 com probabilidade θ_k e recompensa de 0 com probabilidade $1-\theta_k$. Neste trabalho, para cada ação k a função de densidade de probabilidade anterior de θ_k é dada por uma distribuição β com parâmetros $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K)$ e $\boldsymbol{\beta} = (\beta_1, ..., \beta_K)$, conforme a Equação 2.

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\,\Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1} \tag{2}$$

sendo que Γ denota a função *gamma*. A distribuição é atualizada de acordo com a regra bayesiana e o conjugado β anterior, conforme as informações/observações são coletadas. Respectivamente, os valores de α_k ou β_k são incrementados em 1 a cada sucesso ou fracasso observado. Aqui, $p_A(g)$, ou seja, a probabilidade da ação A na geração g, é igual a θ_k e a ação (combinação de heurísticas) selecionada corresponde àquela com maior valor de $p_A(g)$. A probabilidade de sucesso é amostrada aleatoriamente da distribuição posterior.

2.2.2. Heurísticas de Baixo Nível - Recombinação, Busca Local e Mutação

A HH seleciona uma combinação de heurísticas que envolve três categorias: recombinação, busca local e mutação. A recompensa ou *score* de cada heurística é calculada baseando-se na melhoria que determinada heurística trouxe para as soluções candidatas. A recompensa de uma combinação é dada pela soma do *score* de cada heurística que a compõe.

Para a etapa de **reprodução ou recombinação**, situação em que novas soluções ou indivíduos são gerados com base na combinação de dois ou mais indivíduos da população, foram propostas duas heurísticas:

- Recombinação 1: Dois indivíduos da população, definidos como pais, são escolhidos e a partir de genes aleatórios dos dois pais (aleatoriamente dentre os dois em cada iteração) são criadas novas soluções candidatas.
- Recombinação 2: Dois pais são escolhidos e, dentre eles, o que tiver melhor *fitness* recebe o rótulo de privilegiado. As posições do pai privilegiado são percorridas $n/2$ vezes (onde n é a dimensão do problema - número de instalações/localidades) de forma circular a partir de uma posição selecionada aleatoriamente em cada iteração. Durante o percurso são copiadas aleatoriamente as posições i do pai privilegiado para as posições i do filho, mantendo $n/2$ facilidades aleatórias com posições iguais para ambos. As demais posições do filho são preenchidas com valores do pai menos privilegiado, de forma que cada ocorrência de posição vazia do filho recebe o gene do pai menos privilegiado na posição i, caso não gere uma solução infactível. Desta forma, as demais posições do filho são valoradas sem levar em consideração o mesmo índice i para o pai e filho.

A **busca local** é aplicada em um dos cinco melhores indivíduos da população. Essa medida foi tomada de forma a evitar que as heurísticas de busca local ficassem presas a um mínimo local. Três técnicas de busca local foram implementadas, as quais são descritas a seguir:

- Busca local 1: Cada facilidade passa pela troca de posição com todas demais, gerando então o produto cartesiano das instalações dispostas na solução candidata. Após cada troca, avalia-se se esta trouxe uma melhora no *fitness* do indivíduo, caso não, é desfeita a operação;
- Busca local 2: As posições do indivíduo são trocadas de forma que a primeira se torna a última, a segunda instalação troca com a penúltima, e assim sucessivamente. Cada troca é mantida se trouxer uma melhoria ao indivíduo;
- Busca local 3: O algoritmo seleciona uma posição aleatória do indivíduo e então ordena em pares as instalações considerando a posição atual e a próxima, ou seja, a posição i e $i + 1$ são comparadas, se $i + 1$ for maior que i, ambas trocam de posição. Se a troca não trouxer melhoria ao indivíduo ela é desfeita. Isso se repete sucessivamente até que o número de iterações seja igual ao tamanho do indivíduo (n), de forma a manter um número de iterações compatíveis com o tamanho n que varia a cada instância.

O uso dos operadores de **mutação** é baseado em 3 implementações. Em cada um dos casos, se a mutação aplicada não melhorar o indivíduo, a perturbação feita por ela é desfeita e o indivíduo é restaurado à sua forma original. São elas:

- Mutação 1: Efetua a troca de duas posições aleatórias no vetor de instalações do indivíduo;
- Mutação 2: Uma posição aleatória é escolhida no vetor de instalações e denominada *pivô*, outra posição aleatória é escolhida e tem seu valor permutado com àquele da posição anterior ao pivô;
- Mutação 3: Uma posição pivô aleatória é escolhida onde o pivô e os dois elementos anteriores a ele tem sua ordem invertida no vetor de instalações. Por exemplo, um indivíduo $1 - 2 - 3 - 4$ com pivô igual 3 resulta em um indivíduo $3 - 2 - 1 - 4$. Essa é uma mutação mais agressiva que as anteriores citadas, pois altera de posição 3 instalações, e pode gerar saltos maiores na otimização.

No algoritmo proposto, a abordagem utiliza como pais o melhor indivíduo da população e outro aleatório. O número máximo de filhos por heurística é definido pelo parâmetro $MaxF$ (exibido na Tabela 2). Dentre os filhos gerados e os pais, são mantidos apenas os 2 melhores indivíduos dentre eles. Portanto, a HH proposta seleciona dentre 18 possíveis combinações de heurísticas. Cada combinação é composta por uma heurística de cada categoria (recombinação, busca local e mutação).

3. Hiper-heurística proposta para o PQA

A HH proposta neste trabalho, denominada HH$_{TS}$, é baseada na abordagem *Thompson Sampling*. Ela atua ao longo da evolução de um algoritmo memético quando este gera novas soluções. A HH seleciona diferentes combinações de heurísticas pré-definidas divididas em três categorias denominadas: *hRecombinação*, *hBuscaLocal*, *hMutação*. Foram propostas 2 heurísticas de recombinação, 3 heurísticas de busca local e 3 heurísticas de mutação (descritas na Seção 2.2.2) para a geração das 18 combinações que compõem o conjunto de heurísticas de baixo nível para a HH. A Figura 1 apresenta os passos do algoritmo base e o ponto em que a HH atua.

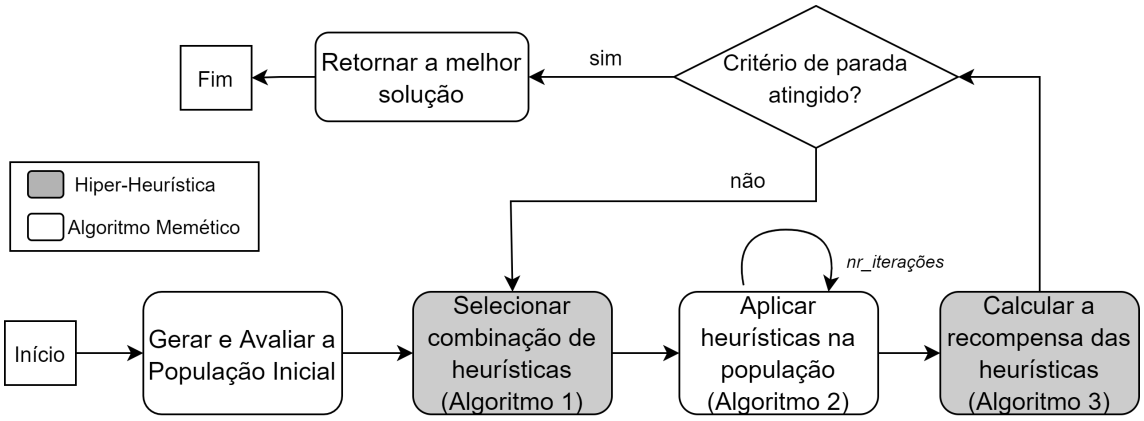

Figura 1. Esquema geral da abordagem proposta - HH$_{TS}$.

No primeiro passo da abordagem proposta ocorre a geração da população inicial, que é feita de maneira aleatória, e a avaliação das soluções (de acordo com a Equação 1). Na sequência, o algoritmo entra em seu laço principal no qual a HH seleciona a combinação de heurísticas a ser aplicada durante $nr\_iterações$ iterações. A seleção da combinação de heurísticas de recombinação, busca local e mutação a ser aplicada é realizada conforme o Algoritmo 1. Após a seleção de heurísticas feita pela HH com *Thompson Sampling* [1], o Algoritmo 2 é responsável por executar $nr\_iterações$ vezes a combinação de heurísticas selecionada. A recompensa representa a métrica de desempenho das heurísticas escolhidas pela HH e, consequentemente, define se a HH fez uma boa escolha de combinações e é calculada conforme o Algoritmo 3. Por fim, o critério de parada é verificado, o qual corresponde a um número máximo de gerações e a melhor solução é retornada.

A aplicação da combinação selecionada acontece conforme o Algoritmo 2, a qual é utilizada ao longo do seu laço principal por $nr\_iterações$ vezes. Neste laço são selecionados os dois pais (o melhor indivíduo da população e outro aleatório) gerando $MaxF$

[1] Implementação baseada em https://pypi.org/project/thompson-sampling/

Algorithm 1 *Thompson Sampling*

Entrada C combinações e número de sucessos e fracassos de cada heurística de baixo nível (inicializados com zero)
Saída Combinação selecionada
1: **para** k $\leftarrow$ 1 **até** C **faça**
2: Amostrar θ_k da distribuição β(Sucesso$_k$+1, Fracasso$_k$+1)
3: **fim para**
4: k = arg max$_k$ θ_k e observe a recompensa () *//cálculo da recompensa conforme Algoritmo 3*
5: Retornar k

filhos. No passo 5 a heurística de reprodução que compõe a combinação selecionada é aplicada na geração de uma nova solução. Este filho é modificado pela heurística de mutação da combinação selecionada (passo 7), de acordo com uma taxa (TxM). Após a conclusão da geração dos filhos, as duas melhores soluções (considerando os pais e os filhos) são mantidas na população. A busca local é aplicada no passo 12 em uma solução selecionada aleatoriamente dentre as cinco melhores da população.

Por fim, a população atualizada é retornada, bem como a recompensa da combinação de heurísticas aplicada.

Algorithm 2 Aplicação das combinações escolhidas

Entrada TxM, hRecombinação, hbuscaLocal, hMutação, pop, MaxF
Saída População atualizada após a geração dos filhos e *feedback* da combinação selecionada
1: i=1
2: **enquanto** $i \leq nr\_iterações$ **faça** *//Controla quantas vezes é utilizada a combinação*
3: Selecionar $pais$
4: **enquanto** !$MaxF$ **faça**
5: $filho$ = hRecombinação($pais$)
6: **se** $rand \leq TxM$ **então**
7: hMutacao($filho$)
8: **fim se**
9: **fim enquanto**
10: Manter os MaxF melhores indivíduos entre $filhos$ e $pais$
11: $indEscolhido$ = Escolher aleatoriamente 1 indivíduo dentre os 5 melhores da pop
12: hBuscaLocal($indEscolhido$)
13: Calcular *score* de hRecombinação, hbuscaLocal, hMutação
14: i = i + 1
15: **fim enquanto**

A recompensa de uma combinação, (conforme os passos do Algoritmo 3) é igual a 1 sempre que a soma dos *scores* das heurísticas que a compõem é maior que zero, caso contrário a recompensa é 0. O desempenho ou *score* de cada heurística é calculado de forma individual para cada categoria, baseando-se na melhoria que determinada heurística trouxe para as soluções candidatas. O cálculo $score_{M+BL}$ para a mutação e para a busca local é dado pela Equação 3. Para a reprodução é considerada a diferença entre a média do *fitness* dos pais e dos filhos, de acordo com a Equação 4, o $score_{REP}$.

$$score_{M+BL} = (original - perturbado) \times original \quad (3)$$

$$score_{REP} = (médiaPais - médiaFilhos) \times médiaPais \quad (4)$$

Algorithm 3 Cálculo da Recompensa

Entrada Soma dos *scores* das heurísticas que compõem a $k-$ésima combinação (SS)
Saída Número de sucessos e fracassos da $k-$ésima combinação

1: **se** $SS > 0$ **então**
2: $Sucesso_k$ = $Sucesso_k$ + 1
3: **senão**
4: $Fracasso_k$ = $Fracasso_k$ + 1
5: **fim se**
6: Retornar $Sucesso_k$ e $Fracasso_k$

4. Resultados e Discussão

A HH_{TS} foi testada em 15 instâncias da série Nug [Nugent et al. 1968]. A matriz de distância destas instâncias contém distâncias de grades retangulares de Manhattan. As instâncias de tamanho n = {14, 16, 17, 18, 21, 22, 24, 25} foram construídas a partir das instâncias maiores excluindo algumas linhas e colunas. A Tabela 1 apresenta as características das instâncias utilizadas nos experimentos deste trabalho, bem como o *fitness* da melhor solução encontrada na literatura. Os parâmetros considerados ao longo de todas as simulações,determinados de forma empírica, são apresentados na Tabela 2.

Tabela 1. Características das instâncias Nug [Nugent et al. 1968].

Instância	n	**Sol**$_{ótima}$
Nug12	12	578
Nug14	14	1014
Nug15	15	1150
Nug16a	16	1610
Nug16b	16	1240
Nug17	17	1732
Nug18	18	1930
Nug20	20	2570
Nug21	21	2438
Nug22	22	3596
Nug24	24	3488
Nug25	25	3744
Nug27	27	5234
Nug28	28	5166
Nug30	30	6124

Foram realizadas 30 execuções independente dos algoritmos. O teste estatístico Kruskal-Wallis, com significância de 0,5, foi aplicado na análise dos resultados. Em todas as tabelas os valores destacados em negrito representam os melhores resultados de *fitness* e as células em cinza claro indicam que há uma equivalência estatística com relação ao método destacado em cinza escuro. Os resultados também são apresentados considerando

Tabela 2. Parâmetros utilizados na abordagem proposta.

Parâmetro	Valor	Descrição
NP	100	Tamanho da população
$MAX - ger$	$n \times 20$	Quantidade máxima de gerações
$MaxF$	2	Máximo de filhos gerados por reprodução
$nr\_iterações$	2	Número de vezes que a combinação selecionada é utilizada
TxM	10%	Taxa de mutação

gap (Equação 5). O *gap* é calculado entre a melhor solução encontrada (sol_{melhor}) pelos algoritmos propostos e a melhor solução conhecida ($sol_{ótima}$), definida na Tabela 1.

$$gap = \frac{(sol_{melhor} - sol_{ótima}) \times 100}{sol_{ótima}} \quad (5)$$

A Tabela 3 monstra a análise estatística na comparação entre diferentes versões da HH usando TS, incluindo informações sobre o melhor *fitness* e o *fitness* médio dentre as execuções, além do *gap*. A primeira versão, HH_{TS}, atua na seleção das 18 combinações de heurísticas. Ela foi capaz de encontrar a melhor solução (*gap* igual a zero) em 7 das 15 instâncias consideradas, como pode ser observado nas primeiras colunas da Tabela 3.

Tabela 3. Melhor, média e GAP com base em 30 execuções independentes para as versões HH_{TS}, HH_{TS-SP} e HH_{TS-SM}. As células em cinza escuro enfatizam os melhores resultados e em cinza claro indicam equivalência estatística ao melhor resultado, de acordo com o teste Kruskal-Wallis.

	Variações da HH com *Thompson Sampling*								
Instância/ Algoritmo	HH_{TS}			HH_{TS-SP}			HH_{TS-SM}		
	Melhor	Média	*Gap*	Melhor	Média	*Gap*	Melhor	Média	*Gap*
Nug12	578	591,60	**0,00**	582	592,20	**0,00**	578	**589,20**	**0,00**
Nug14	1014	1040,13	**0,00**	1014	**1034,47**	**0,00**	1016	1035,93	0,20
Nug15	1150	1182,40	**0,00**	1150	1170,40	**0,00**	1150	**1168,33**	**0,00**
Nug16a	1610	1660,80	**0,00**	1612	1645,87	**0,00**	1610	**1636,60**	**0,00**
Nug16b	1240	1277,00	**0,00**	1240	1269,67	**0,00**	1240	**1268,40**	**0,00**
Nug17	1734	2778,87	0,12	1734	1764,27	0,12	1734	**1759,47**	0,12
Nug18	1936	1996,06	0,31	1930	1973,00	**0,00**	1930	**1965,40**	**0,00**
Nug20	2570	2654,00	**0,00**	2588	2641,60	0,70	2570	**2616,33**	**0,00**
Nug21	2446	2561,73	0,33	2442	2495,73	**0,00**	2438	**2474,67**	**0,00**
Nug22	3604	3724,06	0,22	3596	3647,73	**0,00**	3596	**3641,40**	**0,00**
Nug24	3490	3639,33	0,06	3510	3680,13	0,63	3488	**3546,93**	**0,00**
Nug25	3746	3850,46	0,05	3744	3816,27	**0,00**	3746	**3785,33**	0,05
Nug27	5234	5410,46	**0,00**	5234	5408,93	**0,00**	5246	**5327,00**	0,23
Nug28	5200	5421,6	0,66	5208	5352,60	0,81	5194	**5263,67**	0,54
Nug30	6144	6350,80	0,33	6128	6536,93	0,33	6134	**6232,73**	0,16

Após a análise dos resultados da primeira versão da proposta, foi realizada a comparação entre ela e o algoritmo memético sem a utilização da HH, ou seja, o algoritmo aplica as combinações de forma isolada na geração de todas as soluções. O desempenho de cada combinação de heurísticas é apresentado na Tabela 4, sendo que cada coluna representa o valor de *fitness* médio obtido por duas combinações de heurísticas

(recombinação, busca local e mutação). Por exemplo, a coluna "123" indica a aplicação da heurística de recombinação 1, da heurística de busca local 2 e da heurística de mutação 3, esta mesma coluna traz os resultados da combinação "223". Em destaque estão as melhores médias obtidas. As combinações com piores desempenhos considerando todas as instâncias estão em vermelho, enquanto que as melhores aparecem em verde. O desempenho das combinações foi determinado levando em consideração, além dos valores médios de *fitness*, o melhor valor obtido e o *gap* (por questão de espaço estes últimos valores não são apresentados). Ao comparar a versão HH_{TS}, apresentada na Tabela 3, com as estratégias aplicadas de forma isolada, o teste de Kruskal-Wallis indicou superioridade da HH_{TS} diante de qualquer uma das combinações, demonstrando a eficiência da seleção realizada por ela. Diante da identificação de combinações de heurísticas que se destacaram

Tabela 4. Média dos valores de *fitness* com base em 30 execuções independentes para as diferentes combinações de heurísticas. Valores em destaque enfatizam os melhores resultados.

	Média								
Instância/ Algoritmo	**111**	**112**	**113**	**121**	**122**	**123**	**131**	**132**	**133**
Nug12	690,80	692,66	690,46	695,13	690,53	690,87	689,80	686,73	691,07
Nug14	1208,73	1208,73	1199,86	1211,07	1201,13	1212,60	1204,93	1210,20	1201,00
Nug15	1413,53	1400,46	1402,20	1393,07	1407,13	**1386,40**	1409,67	1394,47	1393,07
Nug16a	**1914,13**	1923,73	1926,33	1917,93	1932,20	1919,47	1923,53	1933,67	1926,00
Nug16b	1520,60	1536,40	1531,60	**1519,87**	1526,87	1523,07	1524,40	1538,00	1532,60
Nug17	2085,73	2080,06	2097,40	2098,80	2084,33	2086,53	**2075,13**	2085,60	2075,87
Nug18	2323,40	2320,66	2328,60	2330,80	2331,60	**2307,93**	2316,93	2333,33	2330,33
Nug20	3112,06	3127,66	3108,40	3116,20	3115,47	3113,47	3092,47	3100,60	3082,40
Nug21	3078,86	3075,06	3074,86	3087,80	3058,73	3066,27	3071,60	3070,93	3071,73
Nug22	4570,93	4574,33	4588,53	4599,20	4562,60	4579,07	4596,00	4572,67	**4535,33**
Nug24	4369,27	4364,40	**4348,73**	4367,80	4371,93	4365,07	4362,67	**4348,73**	4392,20
Nug25	4629,13	4639,53	4629,86	4618,67	4615,40	4620,40	4622,13	4616,40	**4603,93**
Nug27	6559,20	**6532,53**	6548,80	6549,00	6559,53	6525,73	6562,60	6554,60	6534,73
Nug28	6429,66	6413,93	6414,60	6431,53	6430,00	6436,53	6458,67	6413,60	**6406,93**
Nug30	7613,87	7605,87	7596,20	7597,67	7606,13	7604,47	7615,20	7589,40	7629,73
Instância/ Algoritmo	**211**	**212**	**213**	**221**	**222**	**223**	**231**	**232**	**233**
Nug12	695,13	**687,27**	688,80	692,13	687,67	697,00	692,53	689,20	697,80
Nug14	1205,40	1205,27	**1197,67**	1207,00	1211,20	1211,80	1212,67	1213,87	1211,33
Nug15	1401,33	1399,47	1393,13	1400,93	1405,87	1398,33	1404,07	1406,73	1400,87
Nug16a	1922,40	1929,07	1928,53	1930,53	1926,73	1919,60	1924,67	1921,47	1912,53
Nug16b	1530,20	1529,27	1532,40	1530,87	1533,93	1529,73	1520,80	1525,80	1533,60
Nug17	2097,13	2098,60	2084,47	2080,33	2087,33	2083,47	2097,60	2080,80	2089,37
Nug18	2322,40	2322,60	2314,40	2321,73	2328,73	2322,47	2318,87	2323,60	2324,67
Nug20	3101,13	3108,13	3103,47	3124,87	3113,27	3100,20	**3081,13**	3100,80	3114,40
Nug21	3086,47	3067,00	3090,13	3070,73	**3049,33**	3072,20	3060,27	3078,67	3082,47
Nug22	4569,80	4582,67	4537,07	4579,87	4585,20	4549,60	4573,60	4580,13	4596,47
Nug24	4388,07	4352,93	4362,07	4367,07	4370,33	4374,07	4375,27	4374,93	4385,53
Nug25	4613,27	4619,67	4608,53	4652,27	4630,80	4620,07	4652,73	4636,67	4630,80
Nug27	6547,80	6556,80	6550,00	6554,80	6536,00	6565,00	6552,47	6555,27	6559,27
Nug28	6430,53	6434,73	6437,60	6421,60	6432,20	6419,33	6415,33	6442,73	6441,13
Nug30	7597,87	7600,40	7611,20	**7583,60**	7609,47	7602,80	7604,47	7606,60	7617,20

positiva e negativamente, duas novas versões foram testadas e seus resultados são apresentados nas últimas colunas da Tabela 3. São elas: i) HH_{TS-SP} incluindo no conjunto de heurísticas de baixo nível todas as combinações excluindo as piores (marcadas em vermelho na Tabela 4) e ii) HH_{TS-SM} somente considerando as melhores combinações (marcadas em verde na Tabela 4) no conjunto de heurísticas de baixo nível. As duas novas versões apresentam um desempenho melhor do que a primeira versão. A versão HH_{TS-SP} possui um desempenho intermediário com relação à HH_{TS} e HH_{TS-SM}, apre-

sentando a melhor média apenas para uma instância, *gap* igual a zero em sete instâncias e é equivalente à melhor (HH_{TS-SM}) em dez casos.

A versão HH_{TS-SM} obteve a melhor média para todas as instâncias, com exceção à Nug14. HH_{TS-SM} apresentou *gap* igual a zero para 9 das 15 instâncias, sendo equivalente estatisticamente à HH_{TS-SP} para a Nug14. Para as instâncias Nug20, Nug24, Nug28 e Nug30, a versão HH_{TS-SM} é estatisticamente melhor que as outras. Apontada como a melhor versão da HH proposta, HH_{TS-SM} obtém a solução ótima em 9 instâncias e o desvio médio percentual da solução ótima (*gap*), considerando todas as instâncias foi de 8,6%, sendo que os maiores *gaps* foram encontrados para as três maiores instâncias.

Os resultados demonstram que a hiper-heurística *Thompson Sampling* é sensível à qualidade das heurísticas de baixo nível disponíveis para serem selecionadas, visto que os melhores resultados foram obtidos com a versão que considera somente as melhores combinações de estratégias.

5. Conclusões

Neste trabalho foi proposta uma hiper-heurística de seleção baseada na abordagem *Thompson Sampling* em conjunto com um algoritmo memético. Três versões da HH proposta são consideradas: HH_{TS}, HH_{TS-SP} e HH_{TS-SP}. A abordagem *Thompson Sampling* foi utilizada na seleção de uma combinação de três categorias de heurísticas (recombinação, busca local e mutação) para a criação das soluções que compõem a nova geração. Um conjunto de 15 instâncias do *benchmark* Nug foi considerado nos experimentos e os resultados foram analisados de acordo com o teste de Kruskal-Wallis.

Os resultados apontam que todas as versões utilizando HH apresentam melhor desempenho, em termos do valor de *fitness*, do que qualquer uma das heurísticas aplicada de forma isolada ao longo do processo evolucionário. Além disso, as versões que consideram heurísticas de baixo nível de melhor qualidade (HH_{TS-SP} e HH_{TS-SP}) geram resultados superiores aos da versão original (HH_{TS}). Todas as versões utilizando HH propostas podem ser consideradas bem sucedidas, pois evitam que o desenvolvedor/usuário precise escolher quais combinações de heurísticas aplicar diante de um problema. Além disso, a abordagem *Thompson Sampling* é uma técnica simples e que não possui parâmetros associados a serem definidos.

Como trabalhos futuros pretende-se utilizar outras heurísticas de alto nível, tais como algoritmos *Multi-Armed Bandit* que consideram informações contextuais para guiar a seleção de heurísticas. Além disso, o estudo do conjunto de heurísticas a serem selecionadas é um importante tema na pesquisa sobre HHs, que é corroborado pelos resultados deste trabalho. Futuramente também é possível testar o controle adaptativo do tamanho e a composição do conjunto de heurísticas de baixo nível.

Agradecimentos

Os autores agradecem à Unicentro e à Fundação Araucária pelo suporte financeiro parcial à pesquisa.

Referências

Beckman, M. and Koopmans, T. (1957). Assignment problems and the location of economic activities. *Econometrica*, 25:53–76.

Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724.

de Oliveira Neto, L. G. (2017). Um algoritmo imuno-inspirado para o problema quadrático de alocação. Master's thesis, Faculdade de Engenharia Elétrica e de Computação/UNICAMP.

Drake, J. H., Kheiri, A., Özcan, E., and Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405 – 428.

Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction*. John Wiley & Sons Inc, 2 edition.

Knowles, J. and Corne, D. (2003). Instance generators and test suites for the multiobjective quadratic assignment problem. In Fonseca, C., Fleming, P., Zitzler, E., Deb, K., and Thiele, L., editors, *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 2003, Proceedings*, number 2632 in LNCS, pages 295–310. Springer.

Loiola, E. M., de Abreu, N. M. M., and Netto, P. O. B. (2004). Uma revisão comentada das abordagens do problema quadrático de alocação. *Pesquisa Operacional*, pages 73 – 109.

Moreira, D. A. (2011). *Pesquisa operacional: curso introdutório*. Cengage Learning, 2 edition.

Nugent, C. E., Vollmann, T. E., and Ruml, J. (1968). An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.*, 16(1):150–173.

Pillay, N. and Qu, R. (2018). *Hyper-heuristics: theory and applications*. Springer Nature, 1 edition.

Rodrigues, O. A. M. (2017). Aprendizado por reforço baseado em agrupamentos para recomendação na ausência de informação prévia. Master's thesis, Programa de Pós-Graduação em Modelagem Matemática e Computacional/CEFET-MG.

Russo, D. J., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. (2021). A tutorial on thompson sampling.

Sabar, N. R., Ayob, M., Kendall, G., and Qu, R. (2015). A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 45(2):217–228.

Sahni, S. and Gonzales, T. (1976). P-complete approximation problems. *Journal of the Association for Computing Machinery*, 23(3):555–565.

Sucupira, I. R. (2007). Um estudo empírico de hiper-heurísticas. Master's thesis, Instituto de Matemática e Estatística/USP.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294.

Çela, E. (2013). *The Quadratic Assignment Problem: Theory and Algorithms*. Combinatorial Optimization. Springer US.