METAHEURÍSTICA A-TEAM APLICADA AO PROBLEMA DO CORTE GUILHOTINADO BIDIMENSIONAL

Marianny Fidelis de Sousa Mariano
Universidade do Estado do Rio Grande do Norte - UERN
mariannyfidelis@gmail.com

Dario José Aloise
Universidade do Estado do Rio Grande do Norte - UERN
aloisedj@gmail.com

Hugo Alexandre Dantas do Nascimento
Universidade Federal de Goiás - UFG
hadn@inf.ufg.br

Jonathan Lopes da Silva
Universidade do Estado do Rio Grande do Norte UERN
nathan13joker@gmail.com

RESUMO

Vários métodos heurísticos buscam resolver o problema de corte guilhotinado bidimensional (PCGB) utilizando estratégias e características diferenciadas para obter soluções de boa qualidade. O presente trabalho fundamenta-se em uma abordagem metaheurística denominada Time Assíncronos *(Asynchronous Team,* ou *A-Team),* a qual é constituída por um conjunto de agentes trabalhando de maneira autônoma e comunicando-se por meio de memórias compartilhadas. Nessa metaheurística, cada agente incorpora uma estratégia heurística em particular para a resolução do problema em questão. Este trabalho propõe um *A-Team* aplicado ao PCGB combinando uma heurística construtiva e uma metaheurística GRASP, bem como adaptações das mesmas, a fim de alcançar resultados superiores aos encontrados pelas heurísticas isoladamente. Testes computacionais foram realizados com uma classe de instâncias conhecida. A abordagem se mostrou competitiva e promissora quando comparada a outros métodos da literatura.

PALAVRAS CHAVE. Time Assíncrono. Corte Bidimensional. Metaheurística.

Área principal (Corte Bidimensional. Time Assíncrono. Metaheurística)

ABSTRACT

Many heuristic methods for solving the two-dimensional guillotine cutting problem use strategies and differentiated features in order to obtain good quality solutions. This paper is based on the Asynchronous Team (A-Team) approach, which assumes a collection of agents working autonomously and communicating through shared memories. Each agent implements a particular problem-solving method. This paper proposes an A-Team for the just mentioned guillotine cutting problem. The A-Team combines a constructive heuristic and a metaheuristic GRASP with adaptations, aiming to achieve better results than those found by the heuristics alone. Computational tests were performed with a set of benchmark instances. The approach proved promising and competitive when compared to other methods described in the literature.

KEYWORDS. Asynchronous Team. Bi-Dimensional Cutting Problem. Metaheuristic.

Main area (Bi-Dimensional Cutting Problem. Asynchronous Team. Metaheuristic.)

1. Introdução

Os problemas de corte e empacotamento costumam ser de fácil entendimento. O interesse por tais problemas é, em parte, explicado por sua grande aplicabilidade prática, especialmente na indústria, uma vez que a perda de matéria prima influencia diretamente na competitividade e no lucro das empresas, e também por seus desafios teóricos computacionais, uma vez que é um problema de alocação de recursos complexo, cujo tempo necessário para encontrar uma solução aumenta à medida que aumenta a quantidade de peças a serem cortadas de uma mesma chapa. (CHUNG, GAREY e JOHNSON, 1982).

No problema de corte bidimensional, deseja-se atender uma demanda de pedidos (itens) de um determinado material, realizando cortes ortogonais sobre objetos (placas ou chapas) em estoque, maximizando a área de utilização de chapas disponíveis. Uma solução ótima para este problema é encontrar padrões de corte que atendam a demanda, minimizando a área de desperdício e, por conseguinte, maximizando as áreas utilizadas das chapas. Para este problema deve-se ainda respeitar a restrição de corte guilhotinado, de modo que cada corte deve ser feito de uma extremidade a outra da placa, sempre dividindo o pedaço em questão em outros dois novos pedaços.

É importante notar que melhorias no processo de corte (por exemplo, diminuição do desperdício de materiais utilizados como matéria prima) podem apresentar ganhos significativos e representar uma vantagem decisiva na competição entre empresas.

Um número crescente de pesquisadores de diferentes áreas, como computação, engenharia, economia e matemática, estão se dedicando ao estudo de tais problemas, investigando e aplicando diversas técnicas e métodos de resolução como: programação linear, programação dinâmica, *branch-and-bound*, heurísticas e metaheurísticas (GOMES, 2011). Assim há uma grande variedade de algoritmos capazes de resolvê-los e produzir soluções satisfatórias que contribuem de maneira singular para a composição de soluções de boa qualidade.

Para solucionar problemas com estas características altamente combinatórias além da etapa da escolha dos algoritmos, também devem ser consideradas estratégias de resolução, como: selecionar e aplicar apenas um dos algoritmos; aplicar vários algoritmos de estratégias distintas isoladamente e identificar a melhor solução ou uma terceira possibilidade, que consiste em selecionar e combinar os algoritmos escolhidos de tal modo que estes cooperem entre si.

Diante deste contexto, o presente trabalho fundamenta-se em uma abordagem denominada *Asynchronous Team (A-Team)*, a qual é constituída por um conjunto de agentes trabalhando de maneira autônoma e comunicando-se assincronamente por meio de memória (s) compartilhada (s). Cada agente está em um loop aberto e incorpora métodos de resolução particular para o problema em estudo. Com isso, agentes podem agir como coletor, processador e fornecedor de informações.

Essa abordagem possibilita a implementação e combinação de um conjunto de agentes com características e estruturas internas particulares e diferentes, no intuito de trabalhar de forma cooperativa.

O objetivo desta pesquisa é demonstrar que a abordagem *A-Team*, quando aplicada ao problema de corte guilhotinado 2D, produz resultados superiores aos que os agentes, quando executados isoladamente, conseguem alcançar. O restante do artigo está organizado da seguinte maneira. Na Seção 2 é apresentada, uma descrição sobre o problema do corte guilhotinado 2D. Na Seção 3, são apresentados detalhes sobre a utilização da abordagem de time assíncrono para o problema em apreço. E por fim, resultados e conclusões são expostos nas Seções 4 e 5, respectivamente.

2. Problema do Corte Bidimensional Guilhotinado

O problema de corte bidimensional caracteriza-se pelo corte de peças retangulares maiores (objetos/chapas), geralmente de tamanho padrão de largura e comprimento (L, C), para obtenção de peças retangulares menores (itens) com dimensões (l_i, c_i) (como visto na Figura 1), de modo a atender uma demanda d_i, para i = 1, 2, 3, ..., n, respeitando $l_i \leq L$ e $c_i \leq C$. A restrição de guilhotina diz respeito ao corte de uma extremidade a outra do objeto (placa) produzindo, a

cada corte realizado, dois novos retângulos, os quais serão utilizados no processo repetidamente até a obtenção de todas as peças desejadas.

Desta forma, o problema abordado tem as seguintes características: os objetos possuem largura e altura fixa, os itens têm dimensões diferenciadas e podem ser rotacionados em 90°, itens e objetos possuem formas retangulares, os cortes são do tipo guilhotinado sendo o principal objetivo minimizar o desperdício dos objetos utilizados.

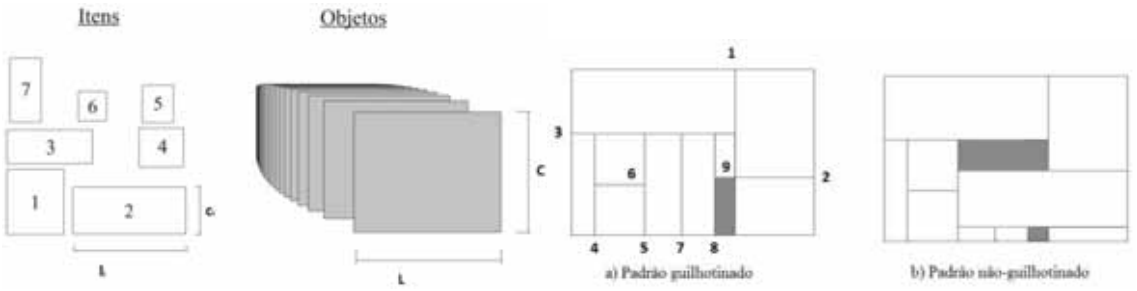

Figura 1: Exemplo de objetos e demanda de itens. Fonte: TEMPONI, 2007

Ainda se tratando do problema de corte, duas formas de visualização de padrões de soluções foram estudadas e investigadas: o *layout* de corte e a árvore binária de corte. A visualização em *layout* de corte, conhecida nas ferramentas de corte, apresenta as soluções na forma de um plano, conforme serão cortadas na placa. A vantagem dessa visualização é o conhecimento da disposição das peças na solução.

O processo de corte guilhotinado, além da representação na forma de *layout* (Figura 2 - a), também pode ser representado na forma de árvore binária (Figura 2 - b), uma vez que a cada corte realizado dois novos pedaços são gerados. A grande vantagem da visualização em árvore binária é que enfatiza a hierarquia dos cortes de uma solução. Isso permite ao usuário ter uma ideia de como todo o processo de corte foi realizado, aumentando seu entendimento sobre o que pode ser feito para melhorar a solução. O processo hierárquico dos cortes, é representado por meio de um conjunto de nós, de modo que os nós intermediários representam um pedaço da chapa cortada (rotulados como "V ou /" e "H ou – "), nomeando cortes verticais e horizontais, respectivamente. Nessa forma de visualização as folhas representam peças ou sobras, como mostrado na Figura 2.

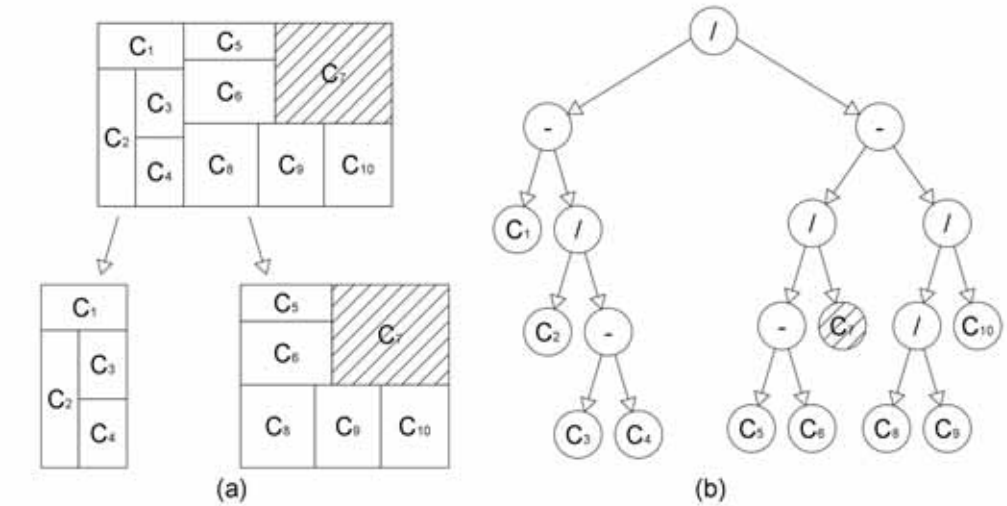

Figura 2: Representação do padrão de corte (a) e sua notação em árvore (b). Fonte: MARINESCU, BAICOIANU, et al., 2009

Diante deste contexto, a união entre as representações em *layout* e árvore se mostrou interessante e adequada ao nosso problema, uma vez que contribuiu para composição de uma estrutura de dados que conseguisse incorporar todas as informações necessárias para execução do algoritmos escolhidos (estratégias de construção e busca local das soluções).

Com isso, além das informações das disposições das peças, sobras e cortes na placa (utilizadas no *layout* clássico), a informação hierárquica do processo de corte que a árvore permite conhecer acrescenta informações adicionais como desperdícios e aproveitamentos de cada sub-região (níveis) da placa (árvore), assim como, permite direcionar o algoritmo para reexecução em sub-regiões não satisfatórias da solução, tudo isso garantindo a restrição de guilhotina. A ideia de reexecução em sub-regiões foi incorporada no processo de busca local (refinamento) de soluções já contidas na memória. Tais algoritmos, ao selecionarem uma solução, utilizam as informações de aproveitamento e desperdício das subárvores de corte que a compõem utilizando a estratégia de desmontar parte da solução e reaplicar algoritmos em sub-regiões específicas. Os detalhes são apresentados na seção que descreve o processo de melhoria.

3. Time Assíncrono aplicado ao Problema do Corte Guilhotinado 2D

Um time assíncrono é constituído por agentes, memórias e fluxo de entrada e saída de informações entre agentes e memórias. A memória é responsável pelo armazenamento de soluções inseridas ou modificadas pelos agentes sendo a única forma de comunicação entre os agentes que compõem um time, ou seja, é compartilhada por todos os agentes. O agente é um algoritmo heurístico responsável pela inserção, aprimoramento ou destruição de soluções na memória.

No *A-Team* proposto (Figura 3) a memória é formada por uma lista de soluções. Cada solução armazena informações úteis a todos os agentes que compõem o time, como por exemplo, uma função de avaliação (FAV) que informa o aproveitamento geral da solução, quantidade total de linhas de corte, somatório total da área das sobras, quantidade de sobras, média das sobras e a lista de placas com os padrões de corte (plano de corte).

Uma vez que a memória é compartilhada entre os agentes, se faz necessária uma padronização. Em vista disso, uma solução é composta por uma lista de planos de corte. E para cada placa, a estrutura que representa o plano de corte da solução é composta por outras três listas: a lista de peças que foram cortadas; a lista de cortes efetuados para obtenção de todas as peças, e a lista de sobras não aproveitadas durante o processo de corte, bem como a posição destas na chapa.

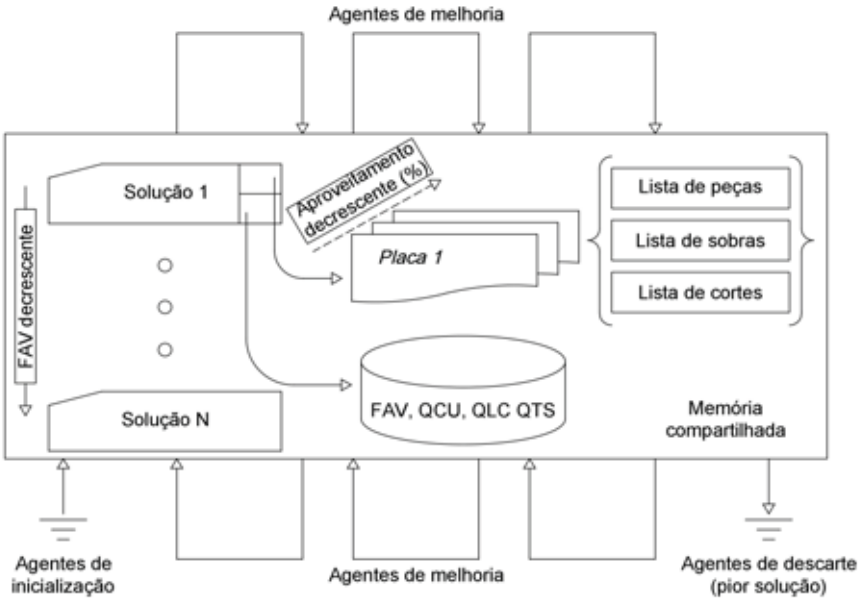

Figura 3: Estrutura do time assíncrono proposto. Fonte: Autoria própria

Com base na estrutura de listas, é possível manter referências suficientemente relevantes para a conversão destas informações para a árvore binária de corte, quando necessária. De modo análogo, a solução seria uma lista de árvores de corte (floresta). Assim, é responsabilidade dos agentes utilizar as informações das listas citadas anteriormente para a codificação e decodificação da estrutura de dados das soluções obtidas da memória.

Para a implementação dos agentes foram selecionadas: uma metaheurística com características gulosa e aleatória e uma heurística construtiva simples para a composição de agentes de construção. Para os agentes de melhoria foram realizadas adaptações na metaheurística e heurística escolhida. Os agentes são baseados nas versões originais dos algoritmos de Velasco (2005) e Nascimento et al. (1999) e adaptações destes para obter agentes com características de melhoria.

3. 1 Agentes de construção e inicialização

O processo de preenchimento da memória é a primeira etapa a ser realizado durante a execução do time. As soluções produzidas nessa etapa são chamadas de soluções iniciais. Para isso são utilizados os agentes de inicialização ou construção. Nas duas subseções a seguir são apresentadas a heurística e a metaheurística selecionadas e incorporadas aos agentes de inicialização.

3.1.1 HHDHeuristic

HHDHeuristic é uma heurística construtiva escolhida para a resolução do problema em estudo apresentada por NASCIMENTO, ALOISE e LONGO (1999). Recebendo como entrada: uma lista de pedidos, a serem atendidos, e o tamanho da placa, o algoritmo funciona da seguinte maneira:

1. Mantém a lista de pedidos ordenada de forma decrescente considerando como critério de comparação as áreas dos pedidos. Em seguida, cria-se uma lista para o armazenamento das sobras existentes (inicialmente vazia).
2. Procura-se cortar a maior peça (pedido) cujas dimensões caibam no menor pedaço disponível (sobra); se a lista de sobras é vazia, considera o tamanho (L, C) da placa.
3. A peça será cortada sempre no canto inferior esquerdo da chapa ou pedaço, de modo que possa ser imediatamente destacada, isto é: pode ser realizado um corte horizontal e em seguida um corte vertical, ou vice-versa. Caso o tamanho da peça seja menor que o pedaço sendo cortado, restarão um ou dois pedaços após a operação.
4. Deve ser escolhido o tipo de corte a ser realizado (horizontal ou vertical) com base no corte que maximiza o aproveitamento da área, não nula, do menor pedaço restante;
5. Caso exista sobras após a operação de corte, estas são adicionadas na lista de sobras, a qual é mantida ordenada de forma crescente pela área da sobra. E repete o processo, retornando ao passo 2.

3.1.2 GRASP-2D

É uma metaheurística gulosa aleatória, adaptada ao problema de corte, caracterizada por partidas múltiplas, onde a cada iteração realiza duas fases: a fase construtiva e a fase de melhoria (RESENDE e SILVA, 2013). A metaheurística utiliza a estratégia de lista de faixas para gerar a solução com os padrões de corte final até a lista de candidatos ficar vazia. E todas as peças sempre são alocadas no canto inferior esquerdo do retângulo escolhido para ser trabalhado.

3.1.2.1 Fase Construtiva

Ao iniciar o algoritmo GRASP devem ser passados como parâmetros a lista de pedidos a serem atendidos, o parâmetro de probabilidade alfa e o número máximo de iterações. Se dados como entrada para o problema de corte, um conjunto $P = \{(c_1, l_1), (c_2, l_2), ..., (c_n, l_n)\}$ de pedidos a serem atendidos, ordenados de modo decrescente por área, e as dimensões da placa retangular R = (C, L), de comprimento C e largura L. A lista de pedidos compõe a lista de candidatos e o

componente alfa determina a composição da lista restrita de candidatos para o qual será selecionado um candidato (pedido) a compor a solução.

Com base no valor atribuído à probabilidade, constrói-se uma Lista Restrita com esses candidatos (LRC), contendo os melhores elementos pertencentes a C. O valor de $\alpha \in [0,1]$ irá determinar o nível de "gula" e aleatoriedade na escolha de peças candidatas. Dessa forma, quanto mais próximo de zero for o valor do α, o algoritmo utilizará inicialmente peças maiores na criação das faixas (será mais guloso na escolha das peças). Em contrapartida, quanto mais próximo do valor um, maior será a chance de escolher qualquer peça aleatória para compor inicialmente as duas faixas.

Em seguida, um candidato (uma peça p_k) é selecionado de forma aleatória na LRC. Se a peça p_k escolhida for maior que a área da faixa a ser preenchida, p_k é excluída da lista de candidatos e a escolha de uma outra peça é feita. Caso contrário, a peça escolhida vai construir inicialmente e temporariamente duas faixas, uma vertical gerada por um corte vertical F_v e uma faixa horizontal gerada pelo corte guilhotinado horizontal F_h, de acordo com as dimensões da peça. Após a inserção da peça escolhida nas faixas (vertical e horizontal) o algoritmo verifica se existem outras peças de mesmas dimensões (demanda) e insere tais peças na faixa atual até que as peças com as dimensões escolhidas extrapolem o limite restante da faixa ou toda sua demanda seja atingida, em seguida a demanda da peça é atualizada

3.1.2.2 Fase de Melhoria

1. Após a etapa de construção a etapa de melhoria é iniciada, onde, para cada faixa temporária (F_v e F_h), são selecionados outros itens na lista de candidatos que caibam nos pedaços restantes das faixas vertical e horizontal (nessa etapa são criadas duas listas de itens, chamadas B_V e B_H, respectivamente). Vale ressaltar que as listas B_V e B_H podem conter itens diferentes;
2. A lista B_V é composta de itens ordenados por largura e refere-se a itens que podem ser encaixados no pedaço restante na faixa vertical.
3. A lista B_H é composta de itens ordenados por altura e refere-se a itens que podem ser encaixados no pedaço restante na faixa horizontal.
4. Após a construção das duas listas B_V e B_H, o algoritmo procura encaixar os itens contidos na lista B_V na Faixa Vertical (F_v) e os itens da lista B_H na Faixa Horizontal (F_h) até ambas as faixas não terem mais espaço suficiente;
5. Após esse processo de encaixe dos itens nas duas faixas temporárias, são calculados os aproveitamentos das duas faixas. A melhor faixa é selecionada (faixa com menor perda interna) e incluída definitivamente no plano de corte. Em seguida:
    a. A lista de candidatos é atualizada, de modo que somente os pedidos pertencentes a faixa escolhida sejam removidos da lista de candidatos;
    b. O processo é repetido, voltando a fase construtiva com a nova lista de pedidos (candidatos). Assim, uma nova faixa será criada a partir do pedaço restante, caso ainda exista; se ainda existir pedidos na lista e as faixas não comportarem, uma nova placa será criada.

3.2 Agentes de melhoria/refinamento

Agentes de melhoria são responsáveis pela seleção de soluções contidas na memória e aplicação de estratégias de melhoria, como busca local. O processo de execução de um agente de melhoria é delineado pela sequência de algumas atividades: ordenação, seleção, execução e inserção da nova solução na memória.

A ordenação enfileira as soluções contidas na memória seguindo algum critério; a seleção escolhe a solução a ser trabalhada seguindo uma política de acesso à memória (políticas de seleção); a execução aplica uma estratégia de melhoria sobre a solução escolhida; por fim, o agente verifica se a quantidade limite de soluções foi atingida, se não, a nova solução é inserida imediatamente; caso a memória já esteja com a capacidade total atingida, então é chamada uma rotina de destruição de soluções.

Na condição da nova solução ser melhor que a pior solução contida na memória e diferente das demais pertencentes à memória, então um agente destruidor é chamado e encarregado de destruir a pior solução. Ao terminar o processo de destruição a nova solução é adicionada a memória.

Entre as políticas disponíveis, em nossa abordagem foram adotadas as políticas de seleção da melhor ou pior solução, e soluções aleatórias contidas na memória. As definições das políticas de seleção são parâmetros configurados dentro de cada agente que irá compor o time.

3.2.1 GRASP-2D - Melhoria

É um agente adaptado da metaheurística GRASP construtiva original. O agente recebe como parâmetros: um alfa, um número de iterações e os parâmetros adicionais de política de seleção que determina qual solução será escolhida pelo agente, como também um valor referente ao aproveitamento das placas que influenciará quais placas serão desmontadas. O funcionamento do agente é determinado pelos seguintes passos:

1. Seleciona uma solução da memória conforme a política de seleção definida;
2. As placas da solução escolhida são ordenadas por aproveitamento de forma decrescente de aproveitamento;
3. O agente desmonta parte da solução escolhida. Para isso, escolhe placas com valor de aproveitamento abaixo do valor definido previamente e estas são desmontadas totalmente;
4. Os itens pertencentes às placas desmontadas tornam-se novamente candidatos no processo heurístico;
5. Aplica o algoritmo GRASP-2D construtivo original, com o alfa, número de iterações, sobre parte da solução (novos itens candidatos);
6. Ao final do processo, a solução parcial gerada (sobre o conjunto reduzido de itens) é unida à que foi mantida (placas que não foram desmontadas);
7. A nova solução gerada é avaliada e os atributos de qualidade são calculados.

3.2.2 HHDHeuristic - Melhoria

Agente que utiliza as vantagens da árvore, antes utilizada de forma interativa (com dicas e ajuda do usuário) agora em um processo totalmente algorítmico e automático. O funcionamento do agente de melhoria acontece seguindo as seguintes etapas:

1. Seleciona uma solução da memória conforme a política de seleção definida;
2. Realiza conversão para uma lista de árvores de corte. Realiza cálculos de dimensões e aproveitamento em cada nó da árvore. As placas da solução escolhida (árvores de corte) são ordenadas por aproveitamento de forma decrescente;
3. Procura por placas com valor de aproveitamento abaixo do valor definido previamente (utiliza informações do nó raiz da árvore), se existir alguma placa com valor igual ou abaixo do especificado essa é desmontada por completo; caso a placa possua um valor superior o agente realiza uma busca em largura na árvore de corte e desmonta subníveis da árvore (sub-regiões) da placa com um valor de aproveitamento igual ou abaixo do valor de aproveitamento de nível especificado;
4. Assim, o agente faz uma varredura para todas as placas (representadas agora como árvore) da solução escolhida, desmontando-as por completo ou parte da placa (nível da árvore). Para isso, utiliza informações do nó raiz e nós intermediários.
5. Ao final do processo as peças antes pertencentes às placas que foram desmontadas voltam a ser pedidos não atendidos. Em caso de desmonte de um nível, uma referência para as dimensões do nó desmontado é inserida na lista de sobras e as peças pertencentes a essa região da placa são inseridas na lista de pedidos.
6. As listas de pedidos e sobras são atualizadas depois do processo de varredura para todas as árvores da solução.
7. Por conseguinte, o algoritmo *HHDHeuristic* é aplicado sobre parte da solução (sobre o conjunto reduzido de itens que foram desmontados) que, primeiramente, verifica se

existe algum pedido, da nova lista de pedidos criada, que caiba nas sobras das placas que não foram desmontadas.

8. Se nenhum pedido couber nas sobras das placas já existentes, o algoritmo se encarrega de criar novas placas até atender toda a demanda de itens restantes;
9. Por fim, a qualidade e atributos da nova solução são novamente calculados.
10. Ao término do processo, é feita novamente a decodificação da árvore para a representação de lista de soluções, e cada solução é representada pelas três listas (peças, cortes e sobras).

3.3 Agente de destruição

Agentes com a função de monitorar, julgar, selecionar e eliminar soluções não promissoras de uma memória sob um critério de avaliação de qualidade das soluções, abrindo espaços para inserção de novas soluções, evitando que memórias recebam quantidades de soluções indesejadas. Para isso, utilizam políticas de destruição. A política de destruição escolhida foi a da pior solução, utilizando como critério a função de avaliação (FAV), de forma que a pior solução é selecionada e removida esta da memória.

3.4 Avaliando a qualidade das soluções

Com o objetivo de avaliar e ordenar as soluções contidas na memória, a cada solução são associadas algumas informações adicionais, são elas: função de avaliação (FAV), quantidade total de chapas utilizadas (QCU), quantidade total de linhas de corte (QLC) e a quantidade total da área das sobras (somadas de todas as chapas) (QTS). Seguindo a ordem dos critérios, quando as soluções possuem a mesma função de qualidade/avaliação (FAV), o critério de desempate será a quantidade de placas (menor número de placas), seguido, da quantidade total de linhas de cortes da solução (menor número), somatório total da área de sobras e a quantidade de sobras (menor número). Tais critérios também são utilizados para diferenciar as soluções.

A função de qualidade (FAV) utilizada como critério de avaliação das soluções pode ser obtida pela seguinte equação:

Equação 1 – Função de avaliação de uma solução

$$FAV = \frac{\left(\sum_{i=1}^{j} (Somatório\ área\ itens\ da\ placa)_i\right) + d_k}{j * C * L}$$

Onde *i* refere-se a uma placa *i* específica entre um total de *j* placas; l_i refere-se a largura do item *i* especificado; *j* refere-se a quantidade de placas total da solução; a variável d_k representa a maior área de desperdício entre todas as placas da solução. E, por fim, *C * L* faz referência a área de uma placa. A variável d_k foi incorporada a *FAV*, visto que por meio desta é possível adicionar mais informação sobre uma solução final. Desse modo, consegue-se melhor distinguir soluções finais com o mesmo número de placas. A distinção ocorre pois, soluções com os mesmos números de placas podem se diferenciar nos arranjos de seus itens; assim, ordenando tais soluções, a última placa passa a ser informação significativa, uma vez que, aquela que possuir maior área de sobra entende-se que na solução conseguiu-se melhor arranjar os itens em placas anteriores.

4. Resultado

Os experimentos computacionais foram realizados sobre uma classe entre dez conhecidas na literatura apresentadas por Lodi et al. (1999) e disponível na OR Library. Cada classe tem cinquenta entradas, dividida em dez instâncias com quantidades *n* de itens diferentes sendo *n = {20,40,60,80 e 100}* itens. As classes se diferem pelo tamanho da placa variando de *10 x 10* à *300 x 300* e quanto a heterogeneidade (diversidade) dos itens. A classe 10 foi escolhida entre as dez classes existentes, por ser a mais heterogênea.

Deste modo, foram testadas 50 entradas, sendo 10 instâncias diferentes para *20* itens, outras 10 para *40, 60, 80 e 100* itens, respectivamente. De forma complementar, para cada uma das entradas foram realizadas rodada de dez execuções, com tempo médio de dez minutos cada.

Para a escolha da configuração de time escolhida foram realizados testes, a priori, a fim de encontrar parâmetros que melhor se ajustassem aos agentes contidos no time e à classe testada.

A estrutura do time assíncrono proposto é constituída por nove agentes distintos e uma única memória. Dois agentes construtores, seis agentes de melhoria e um agente destruidor. Para os dois agentes chamados "GRASP_Construtivo" inicializadores foram atribuídas duas configurações de alfa distintas. Os parâmetros de alpha atribuídos a cada agente foram respectivamente (0.5; 0.75) e 20 soluções geradas pelos agentes objetivando preencher a memória com soluções iniciais. A configuração dos parâmetros dos seis agentes de melhoria do time é apresentada na Tabela 1.

Agentes	Políticas de seleção	Aproveitamento da placa	Aproveitamento Nível	Alfa
HHDHeuristcMelhoria	Melhor Solução	96%	80%	--
HHDHeuristcMelhoria	Solução Aleatória	96%	80%	--
HHDHeuristcMelhoria	Pior Solução	96%	80%	--
Grasp2d_Melhoria	Melhor Solução	96%	--	0.25
Grasp2d_Melhoria	Solução Aleatória	97%	--	0.75
Grasp2d_Melhoria	Pior Solução	97%	--	0.25

Tabela 1: Parâmetros de configuração dos agentes de melhoria. Fonte: Autoria própria

O time para o problema de corte abordado nesse trabalho foi implementado na linguagem Java utilizando a abordagem cliente-servidor, onde os agentes funcionam como clientes (em programas independentes) e as memórias funcionam como servidores, fornecendo serviços (acesso e escrita) aos agentes a ela conectados. Os testes foram executados em um computador Intel Core 2 Duo 2.1GHz, com 6GB de memória RAM, utilizando o sistema operacional Linux Ubuntu 12.04. Os resultados obtidos para os testes computacionais serão apresentados a seguir.

Em particular estabelecemos o tamanho da memória de 50 soluções para os testes com todas as instâncias. Essa relação foi obtida por meio de testes extras a priori e se mostrou suficiente. O aumento no tamanho da memória não apresentou melhorias significativas no time, além do aumento considerável no tempo de execução, já que um controle maior sobre as soluções é exigido.

E para todas as tabelas são exibidos para cada instância, os valores referentes à função de avaliação (FAV) para: a melhor solução, a pior solução, a média e o desvio padrão das soluções encontradas entre as dez execuções testadas. Vale salientar que os valores apresentados na tabela de forma decimal podem ser representados também na forma de porcentagem. Tais resultados são apresentados na Tabela 2.

Em cada tabela (20 e 40 itens) os resultados apresentados são divididos em informações referentes aos construtores no início da execução (1º bloco da tabela – parte superior) e pelo time completo no final da execução (2º bloco – parte inferior). Para o valor de qualidade (FAV) quanto mais próximo de 1, melhor o aproveitamento geral da solução encontrada, sendo que o valor 1 representa padrões de cortes perfeitos em todas as placas.

No intuito de verificar e avaliar o comportamento de sinergia do time proposto, constituído pelos seis agentes, também foram realizados 2 testes extras com duas configurações de time diferenciadas. A primeira configuração se constituiu somente por agentes GRASP de melhoria (três agentes) e a segunda configuração apenas por agentes de melhoria HHDHeuristic (três agentes). Ambos os agentes com os mesmos parâmetros de configuração do time completo composto pelos seis agentes de melhoria. A ideia foi avaliar o comportamento de sinergia do time completo em relação às outras duas configurações.

Para estes testes extras foram escolhidos somente as entradas de (60, 80 e 100 itens). Assim, é possível comparar resultados entre o time completo, o time composto só por GRASP e outro time composto só por agentes HHDHeuristic para as instâncias com 60, 80 e 100 itens.

		Instâncias	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
20 Itens	Construtores	Melhor	0,8781	0,9503	0,9301	0,9218	0,9331	0,9383	0,9138	0,9324	0,9294	0,9336
		Pior	0,7678	0,7433	0,7814	0,7620	0,7287	0,7494	0,7248	0,6859	0,8005	0,7916
		Média	0,8209	0,8564	0,8486	0,8231	0,8351	0,8539	0,8337	0,8388	0,8491	0,8708
		Desv. Pad.	0,0410	0,0536	0,0323	0,0439	0,0399	0,0346	0,0465	0,0424	0,0331	0,0275
	Team Completo	Melhor	0,8781	0,9680	0,9584	0,9349	0,9331	0,9652	0,9380	0,9324	0,9513	0,9719
		Pior	0,8781	0,9625	0,9584	0,9228	0,9331	0,9533	0,9347	0,9145	0,9405	0,9540
		Média	0,8781	0,9649	0,9584	0,9266	0,9331	0,9640	0,9374	0,9270	0,9476	0,9651
		Desv. Pad.	0,0000	0,0028	0,0000	0,0033	0,0000	0,0038	0,0014	0,0086	0,0047	0,0059
40 Itens	Construtores	Melhor	0,9212	0,8969	0,9215	0,9363	0,9237	0,9286	0,9334	0,9135	0,9212	0.8564
		Pior	0,7905	0,7202	0,7754	0,7501	0,7182	0,7067	0,7385	0,7473	0,7569	0,7007
		Média	0,8613	0,8186	0,8572	0,8432	0,8330	0,8369	0,8400	0,8432	0,8485	0,7935
		Desv. Pad.	0,0247	0,0380	0,0392	0,0403	0,0442	0,0350	0,0250	0,0247	0,0348	0,0376
	Team Completo	Melhor	0,9630	0,9303	0,9490	0,9725	0,9607	0,9659	0,9739	0,9569	0,9623	0,9334
		Pior	0,9527	0,9283	0,9395	0,9475	0,9511	0,9518	0,9630	0,9429	0,9494	0,9237
		Média	0,9557	0,9299	0,9453	0,9579	0,9547	0,9623	0,9661	0,9483	0,9570	0,9284
		Desv. Pad.	0,0028	0,0009	0,0034	0,0068	0,0032	0,0040	0,0031	0,0043	0,0040	0,0036
60 Itens	Team Completo	Melhor	0,9391	0,9518	0,9513	0,9752	0,9751	0,9627	0,9701	0,9509	0,9784	0,9691
		Pior	0,9269	0,9440	0,9303	0,9648	0,9644	0,9471	0,9590	0,9353	0,9601	0,9508
		Média	0,9319	0,9487	0,9375	0,9688	0,9710	0,9529	0,9648	0,9421	0,9686	0,9584
		Desv. Pad.	0,0042	0,0023	0,0073	0,0040	0,0033	0,0056	0,0031	0,0060	0,0056	0,0063
		Sinergia	5	7	3	8	6	4	8	6	6	10
	GRASP	Melhor	0,9384	0,9526	0,9595	0,9730	0,9768	0,9593	0,9687	0,9446	0,9715	0,9613
		Pior	0,9266	0,9398	0,9342	0,9590	0,9634	0,9464	0,9543	0,9345	0,9658	0,9425
		Média	0,9325	0,9453	0,9482	0,9653	0,9701	0,9526	0,9619	0,9388	0,9681	0,9499
		Desv. Pad.	0,0040	0,0043	0,0087	0,0046	0,0041	0,0040	0,0042	0,0042	0,0022	0,0057
	HHDHeuristic	Melhor	0,9170	0,9262	0,9133	0,9664	0,9562	0,9188	0,9604	0,9256	0,9607	0,9183
		Pior	0,9013	0,9086	0,9065	0,9571	0,9434	0,9157	0,9532	0,8992	0,9306	0,9092
		Média	0,9054	0,9148	0,9110	0,9616	0,9510	0,9166	0,9564	0,9128	0,9438	0,9145
		Desv. Pad.	0,0054	0,0057	0,0025	0,0031	0,0042	0,0015	0,0022	0,0118	0,0101	0,0030
80 Itens	Tea	Melhor	0,9611	0,9706	0,9767	0,9709	0,9643	0,9506	0,9651	0,9758	0,9736	0,9481

		Pior	0,9423	0,9527	0,9658	0,9609	0,9373	0,9296	0,9426	0,9557	0,9588	0,9352
		Média	0,9515	0,9590	0,9695	0,9658	0,9471	0,9429	0,9496	0,9651	0,9641	0,9402
		Desv. Pad.	0,0064	0,0059	0,0034	0,0037	0,0078	0,0083	0,0067	0,0074	0,0052	0,0043
		Sinergia	5	5	9	7	5	7	6	4	7	3
	GRASP	Melhor	0,9681	0,9661	0,9652	0,9713	0,9592	0,9413	0,9531	0,9745	0,9648	0,9453
		Pior	0,9421	0,9534	0,9566	0,9565	0,9337	0,9341	0,9443	0,9603	0,9519	0,9273
		Média	0,9520	0,9598	0,9610	0,9639	0,9448	0,9378	0,9485	0,9677	0,9581	0,9417
		Desv. Pad.	0,0087	0,0045	0,0025	0,0049	0,0077	0,0026	0,0029	0,0046	0,0040	0,0058
	HHDHeuristic	Melhor	0,9392	0,9524	0,9707	0,9528	0,9439	0,9275	0,9420	0,9566	0,9560	0,9377
		Pior	0,9207	0,9450	0,9601	0,9260	0,9337	0,9207	0,9293	0,9405	0,9414	0,9315
		Média	0,9281	0,9485	0,9648	0,9412	0,9364	0,9239	0,9346	0,9506	0,9480	0,9353
		Desv. Pad.	0,0051	0,0020	0,0031	0,0090	0,0038	0,0021	0,0036	0,0071	0,0046	0,0021
	Team Completo	Melhor	0,9668	0,9595	0,9737	0,9607	0,9636	0,9643	0,9737	0,9457	0,9613	0,9597
		Pior	0,9554	0,9502	0,9602	0,9378	0,9441	0,9369	0,9654	0,9372	0,9311	0,9321
		Média	0,9620	0,9552	0,9651	0,9455	0,9553	0,9513	0,9711	0,9417	0,9434	0,9436
		Desv. Pad.	0,0039	0,0032	0,0039	0,0063	0,0054	0,0074	0,0027	0,0022	0,0100	0,0091
		Sinergia	7	7	7	6	6	3	5	6	1	5
100 Itens	GRASP	Melhor	0,9651	0,9605	0,9676	0,9632	0,9596	0,9595	0,9773	0,9471	0,9656	0,9561
		Pior	0,9531	0,9451	0,9547	0,9286	0,9471	0,9450	0,9639	0,9316	0,9408	0,9315
		Média	0,9589	0,9527	0,9632	0,9429	0,9539	0,9544	0,9698	0,9398	0,9544	0,9444
		Desv. Pad.	0,0040	0,0045	0,0036	0,0097	0,0043	0,0053	0,0051	0,0052	0,0078	0,0098
	HHDHeuristic	Melhor	0,9510	0,9489	0,9585	0,9428	0,9436	0,9370	0,9651	0,9451	0,9391	0,9353
		Pior	0,9418	0,9392	0,9352	0,9260	0,9396	0,9198	0,9497	0,9382	0,9245	0,9213
		Média	0,9456	0,9424	0,9463	0,9334	0,9413	0,9289	0,9559	0,9413	0,9287	0,9279
		Desv. Pad.	0,0030	0,0028	0,0103	0,0062	0,0015	0,0044	0,0047	0,0023	0,0041	0,0048

Tabela 2: Resultados obtidos para a classe de testes

Por fim, a Tabela 3 apresenta comparação dos resultados com outras heurísticas e metaheurísticas da literatura. As colunas 20 a 100 referem-se a diferentes instâncias de dados, de acordo com a quantidade de itens a serem cortados. A última coluna indica o ano de publicação dos métodos. Os valores nas colunas 20 a 100 são a quantidade de total de chapas encontradas nas soluções na otimização. Além do método GRASP publicado em 2005, há outro apresentado em 2010 que apresenta resultados melhores para algumas instâncias. No entanto, o time proposto, conseguiu produzir resultados iguais e, em certos casos, melhores do que os métodos desenvolvidos até o momento, mesmo usando um GRASP menos efetivo. Isso demonstra a sinergia obtida entre as heurísticas que compõem o time.

Método/Nº Itens	20	40	60	80	100	Ano
FBS2	42	75	103	131	163	1999
FC2	41	74	101	130	163	1999
HHDHeuristic	46	76	105	134	166	1999
TS	43	75	104	130	166	2002
HBP	42	74	102	130	160	2003
GLS	42	74	102	130	162	2003
GRASP	41	74	102	130	163	2005
WA	43	74	102	129	159	2009
GRASP	42	74	100	129	159	2010

MS-LGFi	42	74	101	129	160	2012
EA-LGFi	42	74	101	128	160	2012
FFIH	42	73	101	130	161	2013
BFIH	42	73	101	130	160	2013
CFIH	43	73	101	129	159	2013
FFIH + J4	42	73	101	130	161	2013
BFIH + J4	42	73	101	129	160	2013
CFIH + J4	42	73	101	129	159	2013
Time Assíncrono	41	73	100	127	159	2015

Tabela 3: Resultados obtidos para testes com heurísticas e metaheurísticas da literatura

5. Conclusão

Este trabalho demonstrou que *A-Team* apresenta-se como uma ferramenta de grande valor quando aplicados na solução de problemas considerados complexos, como é o caso do PCBG. Também foi demonstrado que *A-Team* permite, com sucesso, a combinação de algoritmos de características construtivas e de melhoria. A estratégia de trabalhar com soluções parciais, aliada a reexecução em sub-regiões da solução se mostrou bastante eficaz, uma vez que os bons arranjos são mantidos e novas combinações podem ser encontradas no espaço de soluções. A metaheurística GRASP se mostrou ideal, uma vez que podemos equilibrar agentes mais gulosos e aleatórios em espaço de busca reduzidos, apresentando grande diversidade nas soluções obtidas.

Analisando-se estes resultados, verifica-se que mesmo com uma implementação reduzida de agentes, os resultados produzidos, na maioria das vezes, se mostraram melhores que os resultados obtidos por outras heurísticas e metaheurísticas isoladamente.

Referências

Alves, J. d. S.; Longo, H. J. Times assíncronos. In: Lopes, H. S.; de Abreu Rodrigues, L. C.; Steiner, M. T. A., editors, Meta-Heurísticas em Pesquisa Operacional, chapter 9, p. 129-144. Omnipax, Curitiba, PR, 1 edition, 2013.

Chung, F. R.; Garey, M. R.; Johnson, D. S. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods***,** v. 3, n. 1, 1982. p. 66-76.

Gomes, J. A. R. Problema de corte bidimensional: Aplicação a um caso real, *Dissertação de Mestrado*. Novembro, 2011.

Marinescu, D.; Baicoianu, A. . The determination of the guillotine restrictions for a rectangular covering model. *WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering.* 2009.

Nascimento, H. A. D. ; Aloise, D. J.; Longo, H. Uma heurística O(mn) para o corte bidimensional guilhotinado. *XXXI SBPO-Simpósio Brasileiro de Pesquisa Operacional*. 1999. p. 1197-1206.

Resende, M. G. D. C.; Silva, R. M. D. A. GRASP: Procedimentos de Busca Gulosos, Aleatórios e Adaptativos. In: **LOPES, H. S.; RODRIGUES, L. C. D. A.; STEINER, M. T. A.** Meta-Heurísticas em Pesquisa Operacional. Omnipax, 2013. p. 1-20.

Souza, P. S., Asynchronous Organizations for Multi-Algorithm Problems. PhD Thesis, Department of Electrical and Computer Engineering of Carnegie Mellon University, 1993.

Talukdar, S., Asynchronous Teams. Technical Report 207, Carnegie Mellon University. Engineering Design Research Center. Department of Electrical and Computer Engineering, 1993.

Talukdar, S.; Baerentzen, L.; Gove, A. & Souza, P. S., Asynchronous Teams: Cooperation Schemes for Autonomous Agents. J. Heuristics, 4(4):295321, 1998.

Temponi, E. C. C. Uma Proposta de Resolução do Problema de Corte Bidimensional via Abordagem Metaheurística, Belo Horizonte - MG. *Dissertação de mestrado.* Dezembro 2007.

Velasco, A. S. GRASP para o Problema de Corte Bidimensional e Restrito. *Dissertação de Mestrado.* Campos dos Goytacazes - RJ, Agosto 2005.