Capítulo 27

Problema de Formação de Células de Manufatura com Roteiros Alternativos e Considerações de Capacidade

Luiz Carlos Abreu Rodrigues*, Adriano Pereira Balau e Tiago Rodrigues Weller

Resumo: O problema de formação de células de manufatura, considerando a disponibilidade de rotas alternativas e da capacidade de processamento das máquinas disponíveis, é um problema de otimização desafiador, uma vez que impõe a solução de um problema em duas fases: *i*) atribuir máquinas às células; e *ii*) atribuir peças (e seu roteiro de fabricação) às células. O objetivo é o de minimizar o processamento extra-celular de tarefas quando os roteiros de fabricação das peças são selecionados. Duas abordagens híbridas, usando busca tabu (BT) e *simulated annealing* (SA), são testadas para atribuir as máquinas às células de manufatura, enquanto o método de *branch-and-bound* é usado para atribuir as peças (e seu roteiro de fabricação) às células.

Palavras-chave: Tecnologia de grupo, Busca tabu, *Simulated annealing*, *Branch-and-bound*.

Abstract: *Cell formation considering the availability of alternative routings and the processing capacity of available machines is a challenging optimization problem, since it imposes the solution of a two-step problem: i) assign machines to cells; and ii) assign manufacturing parts (routings) to cells. The objective is to minimize extra-cellular processing of tasks when routings of parts are selected. The proposed approaches apply tabu search or simulated annealing to assign machines to cells and branch-and-bound to select routings.*

Keywords: *Group technology, Tabu search, Simulated annealing, Branch-and-bound.*

1. Introdução

Aumentos de produção, redução de custos de produtos e melhorias no gerenciamento de estoque de trabalho em processo (WIP) têm sido cada vez mais buscados pelas indústrias. Apesar de os conceitos associadas ao *Just-in-Time* proporem o uso de células de manufatura, sua definição pode ser uma tarefa desafiadora. O método mais utilizado na indústria é tentar associar peças fabricadas que são visualmente semelhantes. A desvantagem principal deste procedimento é a de que, com eles, as células podem ser subutilizadas ou superar sua capacidade de produção. Ou seja, pode não ser uma tarefa fácil compor células de manufatura balanceadas e coerentes baseando-se apenas na semelhança visual entre as peças produzidas. A Tecnologia de Grupo (TG) foi proposta para permitir a formação de células de manufatura, sem ter em conta a semelhança visual de peças fabricadas. A TG visa identificar grupos de peças que compartilham um grupo comum de máquinas (Burbidge, 1975). A TG geralmente se baseia em uma matriz de incidência máquina-componente para identificar grupos de máquinas e componentes (ou peças) que podem ser agrupados em células. Este conceito evoluiu desde a sua primeira proposição e uma extensa literatura sobre a formação de células foi produzida, numa indicação de que diversos procedimentos heurísticos têm sido propostos para solucionar este problema (Irani, 1999; Nsakanda et al., 2006; Caux et al., 2000). De acordo com Irani (1999), alguns dos critérios para a medição da qualidade de formação de células são: *i*) Minimização de operações extra-celulares; *ii*) minimização do carregamento de máquina imposto por operações extra-celulares; e *iii*) minimização do fluxo de produção entre células de manufatura.

Este trabalho destina-se a resolver o problema de formação de células de manufatura com roteiros de produção alternativos e restrições de capacidade de produção. O objetivo consiste em minimizar o número de operações extra-celulares. Mas, não importando o critério de minimização utilizado, a principal dificuldade de qualquer procedimento utilizado para este problema é que a qualidade de uma solução é medida apenas depois que as máquinas são agrupadas. Isto é, este é um problema desafiador já que, normalmente, ele é resolvido em

*Autor para contato: lcar@utfpr.edu.br

dois passos. Primeiro, as máquinas são agrupadas em células de manufatura e, em seguida, no segundo passo, um roteiro de fabricação é selecionado para cada peça. Quando um roteiro é selecionado para uma peça, esta peça é associada a uma célula, tal que ocorra a minimização na quantidade de tarefas extra-celulares. Esta associação permite calcular a função de avaliação (função objetivo).

No presente trabalho apenas uma breve revisão relacionada ao problema de formação de células de manufatura será apresentado. O leitor pode encontrar revisões na literatura para o problema de formação de células e as suas variações em Irani (1999), Nsakanda et al. (2006) e Caux et al. (2000).

Esta revisão destina-se a identificar as principais meta-heurísticas que já foram propostas para resolver o problema de formação de células de manufatura com roteiros alternativos e restrições de capacidade de produção. Uma heurística utilizando programação linear foi proposta por Nagi et al. (1990) para resolver este problema. Eles iterativamente resolvem os dois passos do problema de formação de células como um procedimento de *hill-climbing* de dois passos. Este mesmo problema também foi resolvido usando um algoritmo genético híbrido (Nsakanda et al., 2006) e de *simulated annealing* (Caux et al., 2000; Xambre & Vilarinho, 2003). Mas, em Caux et al. (2000), o *simulated annealing* foi usado apenas para resolver o primeiro passo do algoritmo, agrupando máquinas em células, e *branch-and-bound* foi utilizado para resolver o segundo passo, associando peças às células de manufatura de acordo com a função de avaliação. O algoritmo genético proposto por Nsakanda et al. (2006) utiliza um algoritmo genético para o agrupamento das máquinas em células. Em seguida, a associação das peças às células é realizada utilizando um método de decomposição baseado em Dantzig-Wolfe. Um sistema de colônia de formigas foi utilizado por Prabhaharan et al. (2005) para resolver um problema bastante semelhante em que o objetivo é o de minimizar a variação total na carga das células e o total de movimentos intercelulares.

A busca tabu (BT) foi utilizada por Lei & Wu (2005) e Lei & Wu (2006) para resolver problemas de formação de células multi-objetivo. Ambas as abordagens não consideraram roteiros alternativos e restrições de capacidade de produção. Por conseguinte, o problema pode ser resolvido em uma etapa porque o agrupamento de máquinas em células é suficiente para permitir o cálculo da função de avaliação. Embora o problema resolvido por Lei & Wu (2005) e Lei & Wu (2006) seja mais simples do que com roteiros alternativos e restrições de capacidade de produção, apenas um número limitado de soluções vizinhas foram geradas aleatoriamente em cada iteração.

A programação linear inteira mista, algoritmos gulosos e BT foram usados por Foulds et al. (2006) para tratar o problema de formação de células de manufatura sustentável, com roteiros de fabricação alternativos. O desafio deste problema está em definir quais máquinas comprar, ou não, quando uma nova célula é criada ou quando novas peças serão produzidas. Foulds et al. (2006) apenas trataram problemas de pequenas dimensões.

Para resolver o problema proposto, uma aplicação de BT e de *simulated annealing* (SA) híbridos são apresentados e os resultados são comparados com os obtidos por Caux et al. (2000), que propôs uma abordagem SA híbrida. A aplicação de BT híbrida que é apresentada neste capítulo foi originalmente apresentada em Rodrigues & Weller (2008). Nesta abordagem, a BT é utilizada para resolver o primeiro passo do problema e *branch-and-bound* é usado para resolver a segunda etapa. Este capítulo está organizado da seguinte forma. Na Seção 2 deste capítulo, são apresentadas a descrição e formulação do problema de formação de células com roteiros alternativos de fabricação e restrições de capacidade de produção. A abordagem proposta de BT híbrido, com seu procedimento *branch-and-bound*, é apresentada na Seção 3. Na Seção 4 será apresentada a abordagem de *simulated annealing* híbrida proposta por Caux et al. (2000), que será comparada com a abordagem de BT híbrida proposta neste trabalho. Resultados computacionais e conclusões são apresentados nas Seções 5 e 6, respectivamente.

2. Problema de Formação de Células de Manufatura

O modelo proposto para o problema de formação de células de fabricação com roteiros alternativos e restrições de capacidade de produção é baseado no seguinte conjunto de premissas básicas:

- A produção é estável ao longo do tempo e identificada como quantidades pré-determinadas de peças produzidas dentro de um horizonte de planejamento determinado;
- Cada peça i é obtida após uma sequência O_i de operações;
- Um conjunto de máquinas ($m = 1,...,TM$) está disponível para a produção de todas as peças, sendo que algumas das máquinas são semelhantes;
- O número máximo de máquinas permitidas por célula ($c = 1,...,C$) é NM.

O modelo matemático deste problema de formação de células, que é apresentado a seguir, foi proposto por Rodrigues & Weller (2008). As variáveis deste problema são:

$$x_{mc} = \begin{cases} 1, & \text{se a máquina } m \text{ é alocada à célula } c, \\ 0, & \text{caso contrário;} \end{cases}$$

$$y_{rp} = \begin{cases} 1, & \text{se a peça } p \text{ é produzida pelo roteiro } r, \\ 0, & \text{caso contrário.} \end{cases}$$

A equação 1 é utilizada quando o objetivo do problema é a minimização de operações extra-celulares. Assume-se que a máquina m_1 precede a máquina m_2 no processamento do roteiro de fabricação r. Esta equação indica que quando o roteiro de fabricação r é escolhido para o processamento da peça p (indicado por $y_{rp} = 1$), então a presença das máquinas m_1 (indicado por $x_{m_1c} = 1$) e x_{m_2} (indicado por $x_{m_2c} = 1$) na mesma célula de manufatura c é verificado. Note que uma operação é extra-celular apenas se $x_{m_1c} = 1$ e $x_{m_2c} = 0$. A Tabela 1 apresenta os parâmetros usados para a modelagem do problema proposto de formação de células de manufatura.

Tabela 1. Parâmetros do problema.

Cap_m	capacidade de processamento disponível da máquina m
d_p	demanda durante o horizonte de planejamento da peça p
FM_{mr}	máquina usada pelo roteiro r após o processamento em m
MR_r	conjunto de máquinas usadas pelo roteiro r
RM_m	conjunto de roteiros que usam a máquina m
RP_p	conjunto de roteiros associados à peça p
tp_{mr}	tempo de processamento do roteiro r na máquina m
NM	limite máximo de máquinas que podem ser alocadas numa célula

O problema estudado está sujeito às seguintes restrições. A restrição 2 impõe que qualquer peça p deve necessariamente ser associada a somente um roteiro de fabricação r. A restrição 3 impõe que qualquer máquina m deve ser associada a uma célula de manufatura c. A restrição 4 limita a alocação de roteiros de fabricação de peças de tal forma que não sejam ultrapassadas as capacidades de processamento das máquinas. A restrição 5 define um limite máximo de máquinas associadas a cada célula de manufatura.

$$\text{(Min) Z} = \sum_{p} \sum_{r \in RP_p} y_{rp} \sum_{m_1 \in MR_r} \sum_{c} x_{m_1c}(1 - \sum_{m_2 \in FM_{m_1r}} x_{m_2c}) \tag{1}$$

sujeito a:

$$\sum_{r \in RP_p} y_{rp} = 1 \quad , \forall p \tag{2}$$

$$\sum_{p} d_p x_{mc} = 1 \quad , \forall m \tag{3}$$

$$\sum_{c} x_{mc} \sum_{r \in RP_p, r \in RP_p} tp_{rm} y_{rp} \leq Cap_m \quad , \forall m \tag{4}$$

$$\sum_{m} x_{mc} \leq NM \quad , \forall c. \tag{5}$$

Devido à complexidade computacional deste modelo, proposto por Rodrigues & Weller (2008), e outros modelos propostos na literatura por Nsakanda et al. (2006), Caux et al. (2000) e Xambre & Vilarinho (2003), serão comparadas duas abordagens meta-heurísticas, usando BT e SA para resolver o problema proposto.

3. Busca Tabu Híbrida

Nesta seção é apresentada uma abordagem de BT híbrida para resolver o problema de formação de células de manufatura, considerando-se roteiros de fabricação alternativos e restrições de capacidade de processamento. A abordagem proposta é dividida em duas etapas que são resolvidas iterativamente, sendo que ambas as etapas apresentam procedimentos originais de poda, ou redução do espaço de busca. Na primeira etapa, a busca tabu (BT) é usada para realizar a atribuição das máquinas às células, respeitando o limite máximo NM de máquinas em qualquer célula. Na segunda etapa, um procedimento *branch-and-bound* (BB) é proposto para identificar os roteiros de fabricação que minimizam o número de operações extra-celulares de peças, respeitando a capacidade de processamento das máquinas, Cap_m.

Um aspecto interessante no uso de uma abordagem de BT híbrida para resolver o problema de formação de células de manufatura proposto é que há pouca literatura utilizando BT para resolver estes problemas. Na verdade, a literatura disponível usando BT para resolver os problemas relacionados, como em Lei & Wu (2005) e Lei & Wu (2006), propuseram processos que não exploraram todo o conjunto de soluções vizinhas. De fato, eles geraram aleatoriamente apenas algumas das possíveis soluções vizinhas. Portanto, tornou-se claro, visando tornar eficiente a BT, que é necessário podar o espaço de busca, eliminando soluções vizinhas infactíveis ou não promissoras. Além disto, seria necessário implementar um procedimento eficiente de seleção do roteiro de fabricação das peças, o qual é resolvido usando BB neste trabalho. Caso contrário, a pesquisa pode adquirir um tempo computacional proibitivo.

3.1 Procedimento de busca tabu

O processo híbrido proposto é resolvido usando busca tabu (BT). O leitor interessado encontrará maiores informações sobre esta abordagem em Glover & Laguna (1997) e no capítulo 3 deste livro. Esta meta-heurística explora possíveis soluções sobre a vizinhança de uma solução vigente. Isto se destina a identificar a melhor solução vizinha da solução vigente usando estratégias de movimento predefinidas. Então, a melhor solução vizinha torna-se a nova solução vigente e a busca na vizinhança é reiniciada mais uma vez. Várias estruturas de memória podem ser adotadas de forma a aprimorar a busca e, idealmente, evitar que ela fique presa em pontos ótimos locais. O leitor pode encontrar muitas possíveis estruturas de memória em implementações de BT em Glover & Laguna (1997). Mas, há uma estrutura de memória, chamada de Lista Tabu, que sempre é encontrada na BT. Quando um movimento é realizado e fornece a melhor solução vizinha da solução vigente, este movimento torna-se proibido (ou tabu) por certo número de iterações. Ou seja, este movimento é introduzido na lista tabu, numa tentativa de evitar ciclos na exploração do espaço de busca. Um movimento colocado na lista tabu só será aceito se ele proporcionar a melhor solução encontrada até aquele ponto da pesquisa. Este procedimento é chamado Critério de Aspiração.

A BT é caracterizada por duas situações de busca. A primeira situação é chamada de Intensificação, quando a busca se aproxima de uma região promissora e esta região tende a ser intensivamente explorada. Se for identificado que a busca tornou-se presa em uma região de ótimos locais, a Diversificação, a segunda situação de busca da BT, é executada. Diversificação é obtida gerando-se uma nova solução inicial ou executando-se vários movimentos consecutivos, normalmente aleatórios, que visam mover a busca para longe da região de busca anterior. A busca é reiniciada após a diversificação. O algoritmo de BT implementado é apresentado no Algoritmo 1.

A melhor solução encontrada até então é identificada como s^{best}. No Algoritmo 1, inicialmente, s^{best} é definida na linha 1. A linha 2 do algoritmo impõe que a busca prossiga até que o critério de parada seja atingido. Na linha 3, a solução inicial é gerada aleatoriamente. A solução inicial s será a nova melhor solução s^{best} se o seu valor Z(s) for menor que o valor do melhor atual Z(s^{best}), conforme indicado nas linhas 4 a 6. Na linha 7, prossegue-se a busca enquanto o critério de diversificação não for atingido. Neste algoritmo, o critério de diversificação corresponde a um número de iterações sem encontrar um novo s^{best}. A solução vigente é identificada como s, a melhor solução vizinha é identificada como s'. A rotina de busca na vizinhança (Algoritmo 2) é ativada na linha 8. As Seções 3.2 e 3.3 apresentam as duas etapas da rotina de busca na vizinhança. A cada iteração, nas linhas 9 a 11, é testado se a melhor solução gerada s' é melhor que s^{best}. Na sequência, na linha 12, a melhor solução s' se torna a nova solução vigente s. Na linha 15, retorna-se a melhor solução s^{best} encontrada.

3.2 Geração de soluções vizinhas

Neste trabalho, serão adotados os mesmos movimentos que em trabalhos anteriores sobre o problema proposto, como em Caux et al. (2000) e em Rodrigues & Weller (2008). Serão usados dois movimentos: *i*) inserção é o movimento quando uma máquina é movida de uma célula de manufatura para outra; *ii*) o movimento de troca ou *swap* ocorre quando duas máquinas trocam entre si suas respectivas células de manufatura. Como mencionado anteriormente, a fim de utilizar de forma eficiente a BT, é necessário podar o espaço de busca, eliminando soluções vizinhas infactíveis ou não promissoras. Ao gerar soluções vizinhas, três restrições foram consideradas:

- Como uma restrição de factibilidade, movimentos de inserção podem ser feitos apenas para células com menos do que o limite máximo de máquinas NM;
- Qualquer célula deve permanecer com uma quantidade mínima de máquinas. Esta restrição foi imposta como uma tentativa de evitar soluções vizinhas não promissoras e impor a quantidade mínima de duas máquinas por célula de manufatura;

Algorithm 1 Algoritmo de busca tabu implementado.

1: $s^{best} = +\infty$
2: **while** não satisfez o critério de parada **do**
3: Gerar solução inicial (s)
4: **if** Z(s) < Z(s^{best}) **then**
5: $s^{best} = s$
6: **end if**
7: **while** não satisfez o critério de diversificação **do**
8: Executar o Algoritmo 2 (gerar soluções vizinhas de s) e selecionar o melhor vizinho (s')
9: **if** Z(s') < Z(s^{best}) **then**
10: $s^{best} = s'$
11: **end if**
12: Atualizar a solução vigente: $s = s'$
13: **end while**
14: **end while**
15: Escrever s^{best}

- Um movimento de inserção ou troca de uma máquina m só pode ser feito para outra célula c se houver pelo menos uma outra máquina relacionada com a máquina m na célula c. Duas máquinas são consideradas relacionadas se existir pelo menos um roteiro de fabricação onde ambas as máquinas estão presentes. Este procedimento de poda é baseado no fato de que uma inserção para uma célula sem outras máquinas relacionadas não pode gerar uma solução melhor que a solução vigente. Assim, evitando-se uma solução não promissora, economiza-se tempo de busca.

O Algoritmo 2 realiza a busca na vizinhança da BT. Na linha 1, o valor da melhor solução é definido como $+\infty$. Na linha 4, é avaliado se o movimento de inserção da máquina m para a célula c satisfaz as três restrições apresentadas no início desta seção. Caso o movimento de inserção seja viável, na linha 5, executa-se o Algoritmo 3, que avalia esta solução. Na linha 9, verifica-se se o movimento de troca da máquina m com a máquina m' satisfaz as três restrições apresentadas no início desta seção. Caso o movimento de troca seja viável, na linha 10, executa-se o Algoritmo 3, que avalia esta solução.

Algorithm 2 Busca na vizinhança.

1: Z(s') = $+\infty$;
2: **for** $m = 1$ to M **do**
3: **for** $c = 1$ to C **do**
4: **if** movimento de inserção na célula c satisfaz restrições **then**
5: Executar o Algoritmo 3 (Avaliação da Solução);
6: **end if**
7: **end for**
8: **for** $m' = 1$ to M **do**
9: **if** movimento de troca de células (entre as máquinas m e m') satisfaz restrições **then**
10: Executar o Algoritmo 3 (Avaliação da Solução);
11: **end if**
12: **end for**
13: **end for**

Os Algoritmos 1 e 2 apresentam diferentes partes do algoritmo de BT, onde as máquinas são alocadas às células, realizando os movimentos de inserção e de troca. Por sua vez, o Algoritmo 3 realiza a avaliação de cada solução vizinha gerada, aplicando o BB para definir a associação ótima de cada peça a um roteiro de fabricação. Ao executar o BB, identifica-se o número de operações extra-celulares da solução vizinha analisada.

Na linha 1 do Algoritmo 3 (avaliação da solução) é executado o Algorimo 4, que executa o BB. Na linha 2 do Algoritmo 3 verifica-se se é tabu o movimento da máquina m, que faz parte da solução vizinha. Se a máquina m (ou m', se for um movimento de troca) é tabu, aplica-se o Critério de Aspiração a partir da linha 3. O movimento da máquina m (e m', se for o caso) só será aceito como melhor vizinha s' se este movimento for melhor que a melhor solução encontrada s^{best}. Neste caso, entre as linhas 4 e 7, define-se que esta solução vizinha é a nova melhor solução s^{best} e nova melhor solução vizinha s', atualizando-se os valores da melhor solução Z(s^{best}) e da solução vizinha Z(s'). Na sequência, na linha 8, redefine-se a máquina m (e m', se for um

movimento de troca) como movimento tabu, atualizando-se até quando esta(s) máquina(s) permanecerá(ão) tabu. Se o movimento da máquina m (ou m', se for um movimento de troca) não for tabu, na linha 10, é analisado se a solução vizinha gerada é melhor que a melhor solução vizinha s' encontrada até este momento. Se a solução vizinha gerada for melhor que a solução s', então esta solução vizinha é definida como a nova solução s' na linha 11 e o valor da solução s', Z(s'), é atualizado na linha 12. Na linha 13 é definido que a máquina m (e m', se for um movimento de troca) é um movimento tabu, indicando-se até quando esta(s) máquina(s) permanecerá(ão) tabu.

Algorithm 3 Avaliação da solução.

```
1: Executar o Algorimo 4;
2: if m (ou m', se for um movimento de troca) ϵ lista tabu then
3:    if Z(s^best) > Z(solução vizinha) then
4:       s^best = solução vizinha;
5:       Z(s^best) = Z(solução vizinha);
6:       s' = solução vizinha;
7:       Z(s') = Z(solução vizinha);
8:       Inclui m (e m', se for um movimento de troca) na lista tabu;
9:    end if
10: else
11:    if Z(s') > Z(solução vizinha) then
12:       s' = solução vizinha;
13:       Z(s') = Z(solução vizinha);
14:       Inclui m (e m', se for um movimento de troca) na lista tabu;
15:    end if
16: end if
```

3.3 Associação de peças a roteiros usando BB

Na abordagem de BT híbrida proposta, um procedimento de $branch - and - bound$ (BB) é aplicado sempre que uma solução factível (devido à alocação viável de máquinas às células) vizinha da solução s é produzida. O objetivo desta etapa da BT híbrida é definir a alocação de um roteiro de fabricação para cada peça, visando minimizar o número de operações extra-celulares, mas levando em conta a capacidade de processamento de todas as máquinas. O desafio do BB é que, se não for devidamente implementado, ele pode apresentar um alto custo computacional. A fim de executar atribuições de roteiros às peças, são propostos dois algoritmos que trabalharão conjuntamente. O Algoritmo 4 é responsável pela preparação das informações necessárias ao BB, enquanto o Algoritmo 5 realizará o BB propriamente dito.

Uma vez que as máquinas foram alocadas nas células na primeira etapa da BT, apresentada na Seção 3.2, na linha 3 do Algoritmo 4 identifica-se o número de operações extra-celulares para cada roteiro r da peça p, já que a sua alocação é testada em cada célula c. Assim, identifica-se a célula onde ocorre o menor número de operações extra-celulares para o roteiro r e registra-se o número de operações extra-celulares em $ECmin_r$.

Peças com apenas um roteiro possível não são incluídas na lista $EXPAND_n$, já que a atribuição ótima do roteiro à peça já é conhecido. O objetivo da lista $EXPAND_n$ é favorecer a busca em profundidade no BB. Na linha 6, para qualquer peça p com apenas um roteiro r disponível, atualizar a capacidade disponível de qualquer máquina m associada ao roteiro r (onde $m \epsilon MR_r$). Isto é realizado, na máquina m, subtraindo-se da capacidade disponível Cap_m o tempo de processamento total do roteiro, levando em consideração a demanda da peça p.

Depois que todos os roteiros associados à peça p foram analisados, na linha 11, para cada peça p, identifique a diferença entre os dois menores $ECmin_r$, indicando-a como $diff_p$. Na linha 8, as peças são ordenadas numa lista chamada $EXPAND_n$, onde n é o nó raiz do BB, em ordem decrescente de $diff_p$. Na linha 12, some o menor $ECmin_r$ (para cada peça p). Esta soma é usada como a solução relaxada ou limite inferior do nó raiz do BB. Na linha 14, aplicar o procedimento de busca do BB (Algoritmo 5).

No nó raiz, a lista $EXPAND_0$ contém todas as peças exceto aquelas com apenas um roteiro disponível. A lista $EXPAND_n$ é atualizada a medida que novos nós são gerados na árvore de busca, através da alocação de um roteiro de fabricação a uma peça. Cada nó n terá sua própria lista $EXPAND_n$. Na medida em que a busca evolui, através da geração de novos nós, o procedimento de BB mantém uma lista dos nós abertos, chamada $OPEN$. Nós abertos correspondem a soluções parciais, quando a lista $EXPAND_n$ ainda não está vazia. Inicialmente, quando a lista $OPEN$ é composta apenas do nó raiz, o limite inferior do procedimento de BB é calculado através do Algoritmo 4 e o limite superior da busca (valor da melhor solução completa

Algorithm 4 Preparação do BB.

1: **for** $p = 1$ to P **do**
2: **for** $r \epsilon RP_p$ **do**
3: Identificar a célula onde ocorre o menor número de operações extra-celulares $ECmin_r$ para cada roteiro r da peça p;
4: **if** p com apenas um roteiro disponível **then**
5: **for** $m \epsilon MR_r$ **do**
6: Atualizar a capacidade disponível Cap_m;
7: **end for**
8: **end if**
9: **end for**
10: $diff_p$ = diferença entre os dois menores $ECmin_r$;
11: Limite inferior do BB = Limite inferior do BB + menor $ECmin_r$;
12: Ordenar peças, em ordem decrescente de $diff_p$, numa lista chamada $EXPAND_n$, onde n é o nó raiz do BB;
13: **end for**
14: Aplicar o Algoritmo 5 (executar o BB)

factível encontrada) é definido como $+\infty$. Em problemas de minimização, os nós abertos na lista $OPEN$ são ordenados em ordem crescente do valor da função objetivo (ou função de avaliação), tal que o primeiro nó nesta lista corresponde à solução parcial avaliada como a mais promissora. O leitor encontrará uma explicação detalhada da lista $OPEN$ e do BB em Pearl (1984). O procedimento de busca do BB (Pearl, 1984) é resumido no Algoritmo 5, apresentado a seguir.

Na linha 1 do Algoritmo 5, define-se que o limite superior da função objetivo, chamado *limsup*, será igual a $+\infty$. No Algoritmo 5, serão eliminados da lista $OPEN$ todos os nós cujo valor da solução é maior ou igual a *limsup*. A linha 2 do algoritmo 5 verifica se a lista $OPEN$ está vazia. A busca do BB é encerrada quando a lista $OPEN$ estiver vazia. Se a lista $OPEN$ não está vazia, ramificar o (fazer o *branching* do) primeiro nó da lista $OPEN$. Na linha 3, o primeiro nó da lista $OPEN$ é identificado como No_{pai}. Isto significa que a primeira peça p na lista $EXPAND_{No_{pai}}$, que é identificada como $p1$ na linha 4, será ramificada. Ou seja, na linha 6, será gerado um novo nó, identificado como No_{filho}, para cada roteiro r associado à peça $p1$ (onde $r \epsilon RP_{p1}$).

Na linha 7, remover o primeiro nó da lista $OPEN$ e na lista $EXPAND_{No_{filho}}$ elimine a primeira peça da lista $EXPAND_{No_{pai}}$, mantendo as demais peças. A cada novo nó gerado, na linha 9, atualizar a capacidade disponível das máquinas, identificando-a como $Cap_{m,No_{filho}}$. Se a capacidade de qualquer máquina for excedida, este novo nó será considerado infactível e não será incluído na lista $OPEN$. Caso contrário, calcule o valor da função objetivo para cada novo nó factível associado ao roteiro r da peça p. Na linha 12, o valor estimado da função objetivo do No_{filho} é dada pela soma do valor da solução estimado para No_{pai} mais Δ, onde Δ é a diferença entre o $ECmin_r$ para o roteiro r e a menor $ECmin_r$ para a peça p, como indicado na linha 11. Note que sempre $\Delta \geq 0$.

Na linha 13 é testado se a lista $EXPAND_n$ não está vazia. Se foi criado um nó aberto (solução parcial) factível, na linha 14, ele é inserido e ordenado na lista $OPEN$. Se a lista $EXPAND_n$ está vazia, novas soluções completas (nós fechados) factíveis foram identificadas. No teste da linha 16, se a nova solução completa é factível e o seu valor da função objetivo ($Z(No_{filho})$) é menor do que o limite superior da função objetivo, chamado *limsup*, então, na linha 17, *limsup* é atualizado para o valor da função objetivo desta solução ($Z(No_{filho})$). Esta solução é salva como a melhor solução até então (s'). Na linha 18, é realizado o procedimento de *fathoming*. Ou seja, eliminar qualquer nó aberto (da lista $OPEN$) cujo valor estimado da função objetivo seja maior ou igual ao limite superior.

A maior contribuição do BB apresentado está em propor a utilização da lista $EXPAND_n$. Pretende-se fomentar, tanto quanto possível, uma busca em profundidade. Portanto, parece haver uma poda significativa do espaço de busca do BB, como indicado nos tempos de processamento da abordagem, apresentados nos resultados.

3.4 Exemplo didático

A seguir, apresenta-se um exemplo didático da abordagem de BT híbrida proposta. A Tabela 2 apresenta os dados usados neste exemplo. Há 10 peças diferentes, cada uma com uma demanda definida e um ou mais roteiros de fabricação disponíveis. A cada operação foi associado um tempo de processamento, identificado em unidades de tempo correspondentes a um percentual da capacidade de produção disponível, que é igual a

Algorithm 5 Procedimento de busca do BB.

1: $limsup = +\infty$;
2: **while** Lista $OPEN$ não é um conjunto vazio **do**
3: No_{pai} = primeiro nó da lista $OPEN$;
4: $p1$ = primeira peça na lista $EXPAND_{No_{pai}}$;
5: **for** $r \in RP_{p1}$ **do**
6: Gerar um novo nó No_{filho};
7: Remover o primeiro nó da lista $OPEN$ e a primeira peça na lista $EXPAND_{No_{filho}}$;
8: **for** $m \in MR_r$ **do**
9: Atualizar a capacidade disponível $Cap_{m,No_{filho}}$;
10: **if** No_{filho} factível **then**
11: Δ = diferença entre o $ECmin_r$ para o roteiro r e a menor $ECmin_r$ para a peça $p1$;
12: $\mathrm{Z}(No_{filho}) = \mathrm{Z}(No_{pai}) + \Delta$;
13: **if** lista $EXPAND_{No_{filho}}$ não está vazia **then**
14: Inserir e ordenar No_{filho} na lista $OPEN$ em ordem crescente do valor estimado da função objetivo;
15: **end if**
16: **if** lista $EXPAND_{No_{filho}}$ vazia AND $limsup > \mathrm{Z}(No_{filho})$ **then**
17: $limsup = \mathrm{Z}(No_{filho})$;
18: Realizar o procedimento de $fathoming$;
19: **end if**
20: **end if**
21: **end for**
22: **end for**
23: **end while**
24: Retornar o valor da solução encontrada

um (1) para todas as máquinas. No total, há 9 máquinas e 27 roteiros disponíveis. Na seção 3.1 foi descrita a primeira etapa da BT híbrida, que foi apresentado no Algoritmo 1.

***i*) Iteração 0 (solução inicial da BT):**

Inicialmente, gerou-se a solução inicial (linha 3 do Algoritmo 1), alocando-se aleatoriamente as 9 máquinas em quatro (4) células de manufatura, como indicado na Tabela 3. Para calcular o custo da solução inicial, foi executado o Algoritmo 4, que corresponde à execução de um BB para alocar as peças às células, tal que o número de operações extra-celulares existentes ($ECmin_r$) seja minimizado. A Tabela 4 apresenta, na linha 2 e 3, um resumo do resultado da execução das linhas 3 do Algoritmo 4. Nestas duas linhas, apresenta-se os dois menores $ECmin_r$ encontrados e, entre parênteses, o roteiro responsável por este $ECmin_r$. A Tabela 5 é o resultado da execução das linhas 4 a 9 do Algoritmo 4. Nesta tabela é apresentada a capacidade de todas as máquinas Cap_m após a dedução dos consumos das peças 4 e 10, que só possuem um roteiro de fabricação e, por isto, são as unicas peças alocadas na solução parcial do nó raiz.

No início do BB, a lista $OPEN$ será composta apenas pelo nó raiz: $OPEN = (0)$. A última linha da Tabela 4 apresenta $diff_p$ que é a diferença entre os dois menores $ECmin_r$ para cada peça, que corresponde à execução da linha 10 do Algoritmo 4. Neste algoritmo, na linha 11, calculou-se o limite inferior do BB, que corresponde ao valor da função objetivo estimada para o primeiro nó na lista $OPEN$. Na solução inicial, o limite inferior do BB corresponde ao valor da função objetivo estimada para o nó raiz. Inicialmente, o limite inferior do BB foi obtido somando-se todos os valores do menor $ECmin_r$ para todas as peças, indicado na linha 2 da Tabela 4. Assim, neste exemplo, o limite inferior do BB é igual a seis (6) operações extra-celulares. A linha 12 do algoritmo prevê a definição da lista $EXPAND_0$, relacionada ao nó raiz. Esta lista é gerada ordenando as peças pela ordem decrescente do valor de $diff_p$, apresentado na última linha da Tabela 4. Observe que as peças 1, 3 e 9 apresentam todas $diff_p$ igual a 1. Logo, o ordenamento entre estas três peças pode ser aleatório. Neste exemplo, peças com mesmo valor de $diff_p$ foram ordenadas em ordem crescente (1, 3 e 9). O mesmo critério foi aplicado para as demais peças que apresentaram $diff_p$ igual a zero. Portanto, a lista $EXPAND_0$ será:

$EXPAND_0 = (1, 3, 9, 2, 5, 6, 7, 8)$.

Tabela 2. Dados do exemplo didático.

Peça	Roteiro	Máquina	Tempo de processamento	Demanda
1	R1	M3	0,1	2
		M5	0,2	
	R2	M4	0,1	
		M5	0,2	
2	R3	M1	0,3	2
		M3	0,2	
	R4	M1	0,3	
		M4	0,2	
3	R5	M1	0,1	2
		M3	0,1	
		M5	0,2	
	R6	M1	0,1	
		M3	0,1	
		M5	0,2	
4	R7	M1	0,2	2
		M5	0,1	
5	R8	M2	0,2	2
		M3	0,2	
	R9	M2	0,2	
		M4	0,2	
6	R10	M4	0,1	2
		M8	0,2	
	R11	M4	0,1	
		M7	0,2	
	R12	M4	0,1	
		M9	0,2	
	R13	M3	0,1	
		M8	0,1	
	R14	M3	0,1	
		M7	0,2	
	R15	M3	0,1	
		M9	0,2	
7	R16	M2	0,2	1
		M4	0,1	
		M8	0,4	
	R17	M2	0,2	
		M3	0,1	
		M8	0,4	
	R18	M2	0,2	
		M4	0,1	
		M7	0,4	
	R19	M2	0,2	
		M3	0,1	
		M8	0,4	
	R20	M2	0,2	
		M4	0,1	
		M9	0,4	
	R21	M2	0,2	
		M3	0,1	
		M9	0,4	
8	R22	M2	0,1	3
		M3	0,1	
	R23	M2	0,1	
		M4	0,1	
9	R24	M6	0,3	2
		M7	0,4	
	R25	M6	0,3	
		M8	0,4	
	R26	M6	0,3	
		M9	0,4	
10	R27	M6	0,1	2

1^a iteração do BB:

A última ação do Algoritmo 4, na linha 14, é iniciar a execução do Algoritmo 5, ou seja, iniciar o BB propriamente dito. Portanto, na linha 1 do Algoritmo 5, define-se o valor inicial de *limsup*. No BB, todas as soluções parciais ou nós da lista $OPEN$ serão podadas desta lista se seu valor for maior que *limsup*. Isto ocorre porque a lista $OPEN$ apenas deve conter soluções parciais promissoras, cujo valor, consequentemente, deve ser sempre inferior (num problema de minimização) à melhor solução completa encontrada, cujo valor será atribuído a *limsup*. Como nenhuma solução completa é conhecida no nó raiz (nó 0), $limsup = +\infty$.

Tabela 3. Geração da solução inicial: alocação de máquinas.

	Célula 1	**Célula 2**	**Célula 3**	**Célula 4**
Alocações	M4, M5 e M8	M1 e M2	M3 e M7	M6 e M9

Tabela 4. Menor número de operações extra-celulares de $ECmin_r$.

	Peça									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Menor Custo	0 (R2)	1 (R3)	1 (R6)	1 (R7)	1 (R8)	0 (R10)	1 (R16)	1 (R22)	0 (R26)	0 (R27)
2^o menor Custo	1 (R1)	1 (R4)	2 (R5)		1 (R9)	0 (R14)	1 (R19)	1 (R23)	1 (R24)	
$diff_p$	1	0	1		0	0	0	0	1	

Tabela 5. Capacidade Cap_m após a alocação das peças 4 e 10.

Máquina								
M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
0,6	1	1	1	0,8	0,8	1	1	1

Na sequência do Algoritmo 5, como a lista $OPEN$ não está vazia, na linha 3 é definido que No_{pai} será o primeiro nó da lista $OPEN$. Ou seja, $No_{pai} = 0$. Na linha 4, a primeira peça na lista $EXPAND_0$ é a peça 1. Nas linhas 5 e 6, como a peça 1 tem dois roteiros (R1 e R2), ela gerará dois nós filhos, identificados como 1 e 2. Na linha 7, removeu-se o primeiro nó da lista $OPEN$ e a primeira peça da lista $EXPAND_0$. Haverá uma lista $EXPAND_1$ e outra $EXPAND_2$ para os nós filhos 1 e 2, respectivamente. Assim, obtém-se:

OPEN = ();
$EXPAND_n$ = (3, 9, 2, 5, 6, 7 e 8) , para $n = 1$ e 2.

A linha 9 prevê a atualização da capacidade disponível $Cap_{m,No_{filho}}$ para os dois No_{filho} (1 e 2), sendo que estas informações são apresentadas nas colunas 4 até 13 da Tabela 6 para os dois nós gerados. A fim de facilitar a visualização dos novos nós 1 (associado ao roteiro *R1*) e 2 (associado ao roteiro *R2*) que foram gerados, a primeira linha desta tabela apresenta a informações do nó pai (nó 0). A terceira coluna da Tabela 6 apresenta o valor da função objetivo (Z(No_{filho})) para cada novo nó (roteiro) factível associado à peça 1. Note que, na linha 12, o valor de Z(No_{filho}) é dada pela soma do Z(No_{pai}) (que é o custo do nó 0 nesta Tabela) mais Δ, onde Δ (calculado na linha 11) é a diferença entre o $ECmin_r$ para o roteiro r e a menor $ECmin_r$ para a peça 1. Na Tabela 4 observa-se que $ECmin_r$ têm valores 1 e 0, respectivamente, para os roteiros R1 e R2. Logo, $\Delta = 1$ para R1 e $\Delta = 0$ para R2. Como as listas $EXPAND_1$ e $EXPAND_2$ (verificadas na linha 13 do algoritmo) não estão vazias, os dois nós (1 e 2) são inseridos na lista $OPEN$ na linha 14 do algoritmo. Esta lista é ordenada pela ordem crescente de seus valores da função objetivo (que são indicados entre parênteses nesta lista):

$OPEN$ = (2 (6), 1 (7)).
$OPEN$ = (4 (6), 3 (7), 1 (7)).

Tabela 6. Capacidade $Cap_{m,No_{filho}}$ após a alocação da peça 1.

N^o	**Roteiro**	**Z**	**Máquina**								
			M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
0		6	0,6	1	1	1	0,8	0,8	1	1	1
1	R1	7	0,6	1	0,8	1	0,4	0,8	1	1	1
2	R2	6	0,6	1	1	0,8	0,4	0,8	1	1	1

2^a à 8^a iterações do BB:

A partir da 2^a iteração, as Tabelas 7 e 8 apresentam um resumo das ações tomadas ao seguir os passos do Algoritmo 5. Até a 7^a iteração deste algoritmo, os passos seguidos são idênticos ao seguido na 1^a iteração. Na Tabela 7, a primeira e segunda colunas indicam, respectivamente, o número da iteração do BB e o nó pai dos nós gerados nesta iteração. A terceira coluna apresenta a lista $OPEN$ obtida ao final desta iteração. Observe que o primeiro nó na lista $OPEN$ na iteração i será sempre o nó pai na iteração $i+1$, como indicado nesta tabela e na Tabela 8. Além disto, observe também que, quando dois ou mais nós na lista $OPEN$ têm o mesmo valor estimado para a função objetivo, estes nós podem ser ordenados aleatoriamente. Porém, optou-se por dar preferência nesta lista ao(s) nó(s) com mais peças alocadas. A quarta coluna apresenta a lista $EXPAND_n$ e os nós n gerados nesta iteração (e que são detalhados na Tabela 8). Observe que a primeira peça na lista $EXPAND_n$ na iteração i será sempre a peça a ser inserida na solução na iteração $i+1$, como indicado na Tabela 8.

Tabela 7. Lista $OPEN$ e $EXPAND_n$ da 2^a à 8^a iterações.

Iter.	No_{pai}	**OPEN**	$EXPAND_n$	
2	2	(4 (6), 3 (7), 1 (7))	(9, 2, 5, 6, 7, 8) ,	para n = 3 e 4
3	4	(7 (6), 5 (7), 6 (7), 3 (7), 1 (7))	(2, 5, 6, 7, 8) ,	para n = 5, 6 e 7
4	7	(8 (6), 9 (6), 5 (7), 6 (7), 3 (7), 1 (7))	(5, 6, 7, 8) ,	para n = 8 e 9
5	8	(10 (6), 11 (6), 9 (6), 5 (7), 6 (7), 3 (7), 1 (7))	(6, 7, 8) ,	para n = 10 e 11
6	10	(12 (6), 16 (6), 11 (6), 9 (6), 13 (7), 15 (7), 5 (7), 6 (7), 3 (7), 1 (7))	(7, 8) ,	para n = 12, 13, 14, 15, 16 e 17
7	12	(18 (6), 21 (6), 16 (6), 11 (6), 9 (6), 19 (7), 20 (7), 13 (7), 15 (7), 5 (7), 6 (7), 3 (7), 1 (7))	(8) ,	para n = 18, 19, 20, 21, 22 e 23
8	18	( )	( ) ,	para n = 24 e 25

Na Tabela 8, a primeira, segunda e terceira colunas indicam, respectivamente, o número da iteração do BB, o nó pai dos nós gerados e a peça p que será inserida na solução nesta iteração. A quarta, quinta e sexta colunas apresentam, respectivamente, o nó filho gerado (No_{filho}), o roteiro associado a No_{filho} da peça p que será alocado e o valor da função objetivo (Z) estimado para esta solução. Na sexta coluna, soluções infactíveis são identificadas por "*inf.*". As colunas 7 a 15 da Tabela 7 apresentam, respectivamente, os valores de $Cap_{m,No_{filho}}$. A função objetivo (Z) será infactível sempre que houver alguma máquina m cujo valor de $Cap_{m,No_{filho}}$ for negativo.

Na 8^a iteração, o nó 25 é a primeira solução completa encontrada, já que a lista $EXPAND_n$, na Tabela 7, será vazia ($EXPAND_{25} = (\ )$) pela primeira vez. Assim, como o nó 25 foi a única solução factível desta iteração, o Algoritmo 5 atingiu a linha 17 pela primeira vez, neste exemplo. Assim, o limite superior $limsup$ assumirá o valor $Z(No_{25}) = 6$. E, na linha 18 deste algoritmo, será realizado o procedimento de $fathoming$. Este procedimento implicou na eliminação de todos os nós da lista $OPEN$, já que, do segundo ao último nó da iteração 7, todos os nós têm valor maior ou igual a $limsup$. Logo, o nó 25 representa a solução ótima encontrada, já que a busca do BB (Algoritmo 5) será encerrada quando a lista $OPEN$ estiver vazia. Assim, a solução inicial da busca tabu terá custo de 6 operações extra-celulares, retornando às linhas 4 e 5 do Algoritmo 1, que definirá a solução inicial como a melhor solução encontrada (s^{best}) até então.

Tabela 8. Análise das soluções geradas nas iterações 2 a 8.

Iter.	No_{pai}	peça	No_{filho}	Roteiro	Z	Máquina								
						M1	M2	M3	M4	M5	M6	M7	M8	M9
2	2	3	3	R5	7	0,4	1	0,8	0,8	0	0,8	1	1	1
			4	R6	6	0,4	1	0,8	0,6	0	0,8	1	1	1
3	4	9	5	R24	7	0,4	1	1	0,6	0	0,2	1	1	1
			6	R25	7	0,4	1	1	0,6	0	0,2	1	1	1
			7	R26	6	0,4	1	1	0,6	0	0,2	1	1	0,2
4	7	2	8	R3	6	0,1	1	0,6	0,6	0	0,2	1	1	0,2
			9	R4	6	0,1	1	1	0,2	0	0,2	1	1	0,2
5	8	5	10	R8	6	0,1	0,6	0,2	0,6	0	0,2	1	1	0,2
			11	R9	6	0,1	0,6	0,6	0,2	0	0,2	1	1	0,2
6	10	6	12	R10	6	0,1	0,6	0,2	0,4	0	0,2	1	0,6	0,2
			13	R11	6	0,1	0,6	0,2	0,4	0	0,2	0,6	1	0,2
			14	R12	*inf.*	0,1	0,6	0,2	0,4	0	0,2	1	1	- 0,2
			15	R13	7	0,1	0,6	0	0,6	0	0,2	1	0,8	0,2
			16	R14	6	0,1	0,6	0	0,6	0	0,2	0,6	1	0,2
			17	R15	*inf.*	0,1	0,6	0	0,6	0	0,2	1	1	- 0,2
7	12	7	18	R16	6	0,1	0,4	0,2	0,3	0	0,2	1	0,2	0,2
			19	R17	7	0,1	0,4	0,1	0,4	0	0,2	1	0,2	0,2
			20	R18	7	0,1	0,4	0,2	0,3	0	0,2	0,6	0,6	0,2
			21	R19	6	0,1	0,4	0,1	0,4	0	0,2	0,6	0,6	0,2
			22	R20	*inf.*	0,1	0,4	0,2	0,3	0	0,2	1	0,6	- 0,2
			23	R21	*inf.*	0,1	0,4	0,1	0,4	0	0,2	1	0,6	- 0,2
8	18	8	24	R22	*inf.*	0,1	0,1	- 0,1	0,3	0	0,2	1	0,2	0,2
			25	R23	6	0,1	0,1	0,2	0	0	0,2	1	0,2	0,2

ii) **Iteração 1 da BT:**

Prosseguindo com a BT, após a geração da solução inicial e, como o critério de diversificação não foi atingido neste caso, na linha 8 do Algoritmo 1 é iniciada a busca na vizinhança (Algoritmo 2). Os movimentos de inserção e de troca para este problema possuem três restrições que foram propostas, no início da Seção 3.2, e que visam podar o espaço de busca das soluções vizinhas. Na Tabela 3, todas as máquinas presentes nas células 2, 3 e 4 não podem sofrer um movimento de inserção, porque sua células ficariam com apenas uma máquina, quando o número mínimo de máquinas por célula é igual a dois. Assim, por exemplo, as máquinas nas células 2, 3 e 4 só podem sofrer movimento de troca nesta iteração da BT.

Para a máquina *M1*, observa-se na Tabela 2 que ela está presente na fabricação das peças 2, 3 e 4. Consequentemente, apenas as máquinas *M3*, *M4* e *M5* estão associadas à máquina *M1*. Analisando-se a Tabela 2, identifica-se na Tabela 9 quais máquinas estão relacionadas entre si, ao haver ao menos um roteiro onde elas estão juntas. A partir das informações da Tabela 9, levando-se em consideração às restrições propostas na Seção 3.2, a Tabela 10 apresenta todas as soluções vizinhas geradas nesta iteração da BT.

A primeira coluna da Tabela 10 identifica cada uma das soluções geradas, onde a solução inicial (s) e a melhor solução vizinha encontrada (s') são apresentadas, respectivamente, na segunda e na quinta linhas. A segunda coluna apresenta o valor associado a cada solução (Z) e nas colunas seguintes apresenta-se a alocação das máquinas às células. Devido ao espaço consumido, a execução do BB só foi apresentada para a solução inicial.

4. *Simulated Annealing* (Caux et al., 2000)

O *simulated annealing* (SA) híbrido proposto por Caux et al. (2000) será resumido nesta seção. O SA simula o processo de mudança de estado dos materiais durante sua têmpera ou recozimento. A fim de garantir o carater didático deste capítulo, o algoritmo apresentado nesta seção é mais detalhado que o descrito por Caux et al. (2000), ainda que ambos se comportem igual.

Na primeira linha, são definidos os parâmetros iniciais: temperatura inicial (C) e a probabilidade do movimento de troca (pp). A cada iteração, será definida uma nova temperatura T na linha 17 do algoritmo. Assim, T corresponde à temperatura a cada iteração, enquanto C indica o seu valor inicial. A processo de geração da solução inicial no SA híbrido, na linha 2, é similar ao da BT híbrida apresentada na Seção 3. Na

Tabela 9. Máquinas associadas entre si.

m	**Máquinas associadas à máquina** m
M1	M3, M4 e M5
M2	M3, M4, M7, M8 e M9
M3	M1, M2, M5, M7, M8 e M9
M4	M1, M2, M5, M7, M8 e M9
M5	M1, M3 e M4
M6	M7, M8 e M9
M7	M1, M2, M3, M4, M5 e M6
M8	M1, M2, M3, M4, M5 e M6
M9	M1, M2, M3, M4, M5 e M6

Tabela 10. Busca na vizinhança na 1^a iteração da BT.

		Alocações das máquinas na solução i			
i	**Z**(i)	**Célula 1**	**Célula 2**	**Célula 3**	**Célula 4**
s	6	M4, M5 e M8	M1 e M2	M3 e M7	M6 e M9
1	5	M1, M5 e M8	M4 e M2	M3 e M7	M6 e M9
2	6	M4, M1 e M8	M5 e M2	M3 e M7	M6 e M9
s'	**3**	**M4, M5 e M1**	**M8 e M2**	**M3 e M7**	**M6 e M9**
3	4	M4, M5 e M8	M3 e M2	M1 e M7	M6 e M9
4	5	M4, M5 e M8	M7 e M2	M3 e M1	M6 e M9
5	6	M2, M5 e M8	M1 e M4	M3 e M7	M6 e M9
6	4	M4, M2 e M8	M1 e M5	M3 e M7	M6 e M9
7	5	M4, M5 e M2	M1 e M8	M3 e M7	M6 e M9
8	5	M4, M5 e M8	M1 e M3	M2 e M7	M6 e M9
9	4	M4, M5 e M8	M1 e M7	M3 e M2	M6 e M9
10	7	M4, M5 e M8	M1 e M6	M3 e M7	M2 e M9
11	6	M4, M5 e M8	M1 e M2	M3 e M7	M6 e M9
12	6	M3, M5 e M8	M1 e M2	M4 e M7	M6 e M9
13	8	M4, M3 e M8	M1 e M2	M5 e M7	M6 e M9
14	8	M4, M5 e M3	M1 e M2	M8 e M7	M6 e M9
15	6	M4, M5 e M8	M1 e M2	M6 e M7	M3 e M9
16	8	M6, M5 e M8	M1 e M2	M3 e M7	M4 e M9
17	6	M5 e M8	M4, M1 e M2	M3 e M7	M6 e M9
18	8	M5 e M8	M1 e M2	M4, M3 e M7	M6 e M9
19	8	M5 e M8	M1 e M2	M3 e M7	M4, M6 e M9
20	6	M4 e M8	M5, M1 e M2	M3 e M7	M6 e M9
21	6	M4 e M8	M1 e M2	M5, M3 e M7	M6 e M9
22	6	M4, M5 e M7	M1 e M2	M3 e M8	M6 e M9
23	6	M4, M5 e M8	M1 e M2	M3 e M9	M6 e M7

linha 3, a busca prosseguirá até que o critério de parada seja atingido. O critério de parada pode ser um número máximo de iterações ou um valor da temperatura T.

Na linha 4 gera-se a solução vizinha s'. Enquanto a BT híbrida preve o teste de todas as soluções vizinhas promissoras (que satisfazem as restrições nos movimentos de inserção e de troca), será gerada apenas uma solução vizinha por iteração no SA híbrido. A cada iteração gera-se um número aleatório entre 0 e 1. Se este número aleatório for menor que a probabilidade de movimento de troca (pp), nesta iteração ocorrerá um movimento de troca. Senão, será realizado um movimento de inserção. Igual à BT híbrida proposta, Caux et al. (2000) propuseram que o SA realize a alocação das máquinas às células e que um algoritmo de BB identifique qual roteiro fabricará cada peça, definindo a solução de menor custo.

Na linha 5 calcula-se o valor de Δ, que é a diferença entre os valores das soluções vizinha (s') e vigente (s). Na linha 6, se Δ for menor ou igual a zero, num problema de minimização, a solução vizinha (s') será melhor ou igual que a solução vigente (s). Assim, s' se tornará a nova solução vigente (s) na linha 7. Posteriormente, nas linhas 8 a 10, verifica-se se esta é a melhor solução encontrada até esta iteração (s^{best}). Se Δ for maior que zero, s' é uma solução pior que s. O SA prevê que haja uma chance igual a exp(- Δ/T) de aceitação

Algorithm 6 SA híbrido proposto por Caux et al. (2000).

1: Definir os parâmetros temperatura inicial (C) e probabilidade do movimento de troca (pp);
2: Gerar solução inicial s;
3: **while** Não atingiu o critério de parada **do**
4: Gerar solução s' vizinha da solução vigente s;
5: Δ = Z(s') - Z(s);
6: **if** $\Delta \leq 0$ **then**
7: Nova solução vigente: $s = s'$;
8: **if** Z(s) < Z(s^{best}) **then**
9: $s^{best} = s$
10: **end if**
11: **else**
12: Gerar um número aleatório r, onde $0 < r < 1$;
13: **if** $r <$ exp(- Δ/T) **then**
14: Nova solução vigente: $s = s'$;
15: **end if**
16: **end if**
17: Atualizar a temperatura T;
18: **end while**
19: Escrever a melhor solução encontrada s^{best}.

desta solução pior s' como nova solução vigente s. O propósito desta chance de aceitação de uma solução pior que s é o de permitir que a busca não fique presa num ponto ótimo local. Se esta chance não existisse, este algoritmo equivaleria a um $hill - climbing$. A chance de aceitação de uma solução pior que s é executada entre as linhas 12 e 15. Ao fim da busca, quando o critério de parada for atingido, na linha 18, escreve-se a melhor solução encontrada s^{best}.

5. Resultados

A abordagem proposta foi executada em um Pentium IV PC com CPU de 3.00 GHz e memória RAM de 2.00 GB. Esta abordagem foi implementada usando Visual Studio 2005. A instância usada foi publicada por Nagi et al. (1990), que a usou para testar sua abordagem proposta, já que ela foi usada também por Caux et al. (2000) para testar sua abordagem de SA híbrido. Os resultados da BT híbrida proposta é comparada aos resultados obtidos por Caux et al. (2000). Nesta instância, 20 peças, com 51 roteiros e 20 máquinas foram agrupadas em 5 células. A solução inicial foi obtida aleatoriamente.

A Tabela 11 indica os resultados apresentados por Caux et al. (2000), em número de iterações. Para testar o SA, dois parâmetros foram variados por teste: temperatura inicial (C) e probabilidade de troca ou *swap* (pp). Ou seja, um vizinho é criado usando probabilidade de troca igual a pp e usando probabilidade de inserção igual a (1 - pp). Para cada conjunto de parâmetros C e pp, 25 execuções foram realizadas. No melhor caso, a solução ótima obtida por Caux et al. (2000) ocorreu, em media, após 218 iterações (para C = 10 e probabilidade de troca pp = 0,5). Os testes com o algoritmo apresentado por Caux et al. (2000) foram consistentes com aqueles apresentados na Tabela 11, já que o desvio padrão da média (quando a média era inferior a 600 iterações) variou entre 199,94 e 333,76 iterações. A Tabela 12 apresenta uma comparação entre os resultados após 100 execuções das duas abordagens (BT e SA). O único parâmetro que foi definido para a BT híbrida foi o tamanho da lista tabu, que foi fixado em cinco (5) iterações devido à dimensão da instância proposta. Para o problema proposto por Nagi et al. (1990), pode ser observado que a abordagem proposta de BT híbrida supera os melhores resultados apresentados no SA híbrido, proposto por Caux et al. (2000). O pior caso nos tempos de CPU foi utilizado para comparar os tempos de processamento de ambas as abordagens por causa do desvio padrão significativo apresentado pelo SA. De fato, uma desvantagem importante do SA foi o seu desvio padrão.

Os autores deste trabalho acreditam que a principal razão para o uso do SA, como visto na revisão da literatura, para o problema proposto está na simplicidade desta abordagem, já que o problema foi resolvido em dois passos. Além disto, devido à sua complexidade computacional, o uso do BB no segundo passo da solução do problema proposto parece exercer uma influência negativa à adoção de BT e algoritmos genéticos, que exploraram várias soluções vizinhas a cada iteração. Ou seja, se o procedimento de BB não for eficaz, torna-se difícil combinar o BB com abordagens como a BT ou algoritmos genéticos.

No algoritmo de BT híbrido implementado, note que três regras foram usadas para pesquisar soluções vizinhas no primeiro passo da abordagem, mas somente a primeira foi uma restrição do problema. As duas

Tabela 11. Número total médio de iterações necessárias para atingir a solução ótima (Caux et al., 2000).

pp	*C*				
	1	**5**	**10**	**15**	**20**
0	2000	556	321	308	1538
0,25	1031	397	450	743	1622
0,5	775	319	218	949	1100
0,75	764	280	671	1861	2000
1	2000	2000	2000	2000	2000

Tabela 12. Resultados obtidos para encontrar a solução ótima usando BT e SA (no seu melhor desempenho).

Abordagem	**Número de iterações**	**Pior tempo de CPU**
BT híbrida	12.7 ± 3.45	2 segundos
SA híbrido (Caux et al., 2000)	218 ± 199.94	5 segundos

regras de poda que foram adicionados ao primeiro passo contribuíram para reduzir o espaço de soluções vizinhas. A segunda contribuição importante deste trabalho foi a incorporação da lista $EXPAND_n$ ao BB, o que tornou o tempo de processamento do algoritmo razoável. Um comentário sobre o exemplo utilizado, proposto por Nagi et al. (1990), é necessário. Foi o único exemplo encontrado para o problema proposto e que tinha sido utilizado em comparações anteriores entre diferentes abordagens. A literatura sobre o problema de formação de células indica o uso de problemas de tamanho similar. Apesar disto, o exemplo usado foi considerado pequeno. Devido a isto, a BT híbrida convergiu bastante rápido para a solução ótima, tornando desnecessária a diversificação para este exemplo. Uma vez que os resultados foram promissores, exemplos maiores serão utilizados no futuro.

6. Conclusões

Este capítulo é focado na solução do problema de formação de células, com roteiros alternativos e considerações de capacidade. Um modelo matemático foi apresentado para este problema. Este é um problema desafiador que requer dois passos para resolver uma iteração usando meta-heurísticas: *i*) atribuição das máquinas às células de manufatura; e *ii*) cálculo da função objetivo. Devido a isto, apenas uma gama limitada de meta-heurísticas tem sido proposta para resolver o problema. Uma abordagem de BT híbrida foi apresentada com duas contribuições importantes: *i*) duas regras de podar para o primeiro passo do problema; e *ii*) um procedimento rápido BB para resolver a segunda etapa do problema. Resultados da BT híbrida superaram os do SA híbrido, proposto por Caux et al. (2000). Estes são os primeiros resultados da abordagem de BT híbrida proposta. Exemplos de maior tamanho serão propostos no futuro, tornando possível explorar melhor outros recursos da BT, como outros mecanismos de memória (Glover & Laguna, 1997). Além disto, outras abordagens, como algoritmos genéticos, podem ser testadas no futuro, utilizando o procedimento BB proposto neste trabalho.

Referências

Burbidge, J.L., *An Introduction to Group Technology*. New York, USA: John Wiley & Sons, 1975.

Caux, C.; Bruniaux, R. & Pierreval, H., Cell formation with alternative process plans and machine capacity constraints: A new combined approach. *International Journal of Production Economics*, 64(1-3):279–284, 2000.

Foulds, L.R.; French, A.P. & Wilson, J.M., The sustainable cell formation problem: Manufacturing cell creation with machine modification costs. *Computers & Operations Research*, 33(4):1010–1032, 2006.

Glover, F. & Laguna, M., *Tabu Search*. Norwell, USA: Kluwer Academic, 1997.

Irani, S.A., *Handbook of Cellular Manufacturing Systems*. New York, USA: John Wiley & Sons, 1999.

Lei, D. & Wu, Z., Tabu search-based approach to multi-objective machine-part cell formation. *International Journal of Production Research*, 43(24):5241–5252, 2005.

Lei, D. & Wu, Z., Tabu search for multi-criteria manufacturing cell design. *International Journal of Advanced Manufacturing Technology*, 28(9-10):950–956, 2006.

Nagi, R.; Harhalakis, G. & Proth, J.M., Multiple routeings and capacity considerations in group technology applications. *International Journal of Production Research*, 28(12):2243–2257, 1990.

Nsakanda, A.L.; Diaby, M. & Price, W.L., Hybrid genetic approach for solving large-scale capacitated cell formation problems with multiple routings. *European Journal of Operational Research*, 171(3):1051–1070, 2006.

Pearl, J., *Heuristics: intelligent search strategies for computer problem solving*. Reading, USA: Addison-Wesley, 1984.

Prabhaharan, G.; Muruganandam, A.; Asokan, P. & Girish, B.S., Machine cell formation for cellular manufacturing systems using an ant colony system approach. *International Journal of Advanced Manufacturing Technology*, 25(9-10):1013–1019, 2005.

Rodrigues, L.C.A. & Weller, T.R., Cell formation with alternative routings and capacity considerations: A hybrid tabu search approach. In: Gelbukh, A. & Morales, E.F. (Eds.), *MICAI 2008: Advances in Artificial Intelligence*. Heidelberg, Germany: Springer-Verlag, v. 5317 de *Lecture Notes in Computer Science*, p. 482–491, 2008.

Xambre, A.R. & Vilarinho, P.M., A simulated annealing approach for manufacturing cell formation with multiple identical machines. *European Journal of Operational Research*, 151(2):434–446, 2003.

Notas Biográficas

Luiz Carlos Abreu Rodrigues é graduado em Engenharia Mecânica (UNICAMP, 1989), mestre e doutor em Engenharia Elétrica (UNICAMP, 1996 e 2000, respectivamente). Atualmente é Professor Associado do Departamento Acadêmico de Mecânica da Universidade Tecnológica Federal do Paraná (UTFPR). Ele tem interesse na área de otimização combinatória, com aplicações em problemas de planejamento, projeto e controle da produção.

Adriano Pereira Balau é graduado em Engenharia de Produção Mecânica (USP/São Carlos, 1999). Atualmente é discente de mestrado no Programa de Pós-Graduação em Engenharia de Mecânica e de Materiais da Universidade Tecnológica Federal do Paraná (UTFPR).

Tiago Rodrigues Weller é graduado em Tecnologia Mecânica (UTFPR, 2004) e mestre em Engenharia Mecânica (UTFPR, 2008). Atualmente é Professor Adjunto do Departamento Acadêmico de Mecânica da Universidade Tecnológica Federal do Paraná (UTFPR). Ele tem interesse em problemas de planejamento, projeto e controle da produção.