MOEA/D com Busca Local para o *Flow Shop* Multiobjetivo

Geovani Ferreira Antunes[2], Sandra Mara Venske[1], Carolina Paula de Almeida[1], Myriam Delgado[2], Ricardo Lüders[2]

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

[2]Pós-Graduação em Engenharia Elétrica e Informática Industrial – Universidade Tecnológica Federal do Paraná (UTFPR) Curitiba – PR – Brasil

geovanyantunes@hotmail.com

Abstract. *The flow shop is a combinatorial problem for which a set of tasks should be sequentially processed by a set of machines. If the sequence of tasks is the same for all machines, then it is called permutation fow shop (PFSP) which is frequently found in assembly lines of industrial plants. This work considers the multiobjective PFSP that minimizes makespan and tardiness. The makespan is the time interval necessary to complete all tasks in all machines and tardiness is the time delay experienced by a task for a given duedate. In this work we consider a multi-objective framework called MOEA/D-DRA (Multi-objective Evolutionary Algorithm based on Decomposition with Dynamic Resource Allocation). In additon, a NEH heuristic is included into a local search embeded into MOEA/D-DRA (differently from other approaches which consider NEH only for the initial population. Experiments have been made for 11 instances of PFSP with 20 to 200 tasks and 5 to 20 machines. The obtained results are compared with those of MOEA/D-DRA using NEH only at the begining. These results show that MOEA/D-DRA with no local search outperforms the proposed approach only for two instances.*

Resumo. *O flow shop é um problema combinatorial em que um conjunto de tarefas devem ser processadas sequencialmente em um conjunto de máquinas. Se a sequência de tarefas é a mesma para todas as máquinas, tem-se o flow shop de permutação (PFSP), frequentemente encontrado em linhas de montagem de plantas industriais. Este trabalho considera o PFSP multiobjetivo que minimiza "makespan" e "tardiness". O "makespan" é o tempo necessário para processar todas as tarefas em todas as máquinas e o "tardiness" é o atraso para finalizar uma tarefa em um dado prazo. A plataforma multiobjetivo utilizada é o MOEA/D-DRA ("Multi-objective Evolutionary Algorithm based on Decomposition with Dynamic Resource Allocation"). Para tanto, é utilizada a heurística (NEH) em uma busca local acoplada ao MOEA/D-DRA (diferentemente de outros trabalhos que utilizam NEH apenas na inicialização). Experimentos com esta abordagem proposta são realizados para 11 instâncias do PFSP com 20 a 200 tarefas e 5 a 20 máquinas. Os resultados obtidos são comparados com os do MOEA/D-DRA utilizando o mecanismo NEH apenas na inicialização da população. Estes resultados mostram que a versão do MOEA/D-DRA sem busca local supera a abordagem proposta em apenas duas instâncias.*

1. Introdução

O *Flow Shop* é considerado um dos problemas de alocação e sequenciamento mais estudados na literatura de Pesquisa Operacional que modela vários sistemas reais da indústria [Baker e Trietsh 2009]. O problema consiste de um conjunto de tarefas a serem processadas por diferentes máquinas, cuja solução envolve a determinação da sequência de tarefas em cada máquina [Yenisey e Yagmahan 2014]. Quando a sequência de tarefas é a mesma para todas as máquinas, o problema é denominado *Flow Shop* de Permutação (PFSP). Este problema é NP-difícil para mais de duas máquinas [Garey et al. 1976]. Portanto, muitas pesquisas focam na implementação de métodos de aproximação, geralmente heurísticas ou metaheurísticas para a solução do *Flow Shop* de Permutação [Fernandez-Viagas et al. 2017].

Assim como diversos problemas reais que possuem objetivos conflitantes, o processo de produção na indústria também pode ser considerado um Problema de Otimização Multiobjetivo (POM) [Zhou et al. 2011]. No processo industrial deve-se minimizar o tempo total das atividades, maximizando a eficiência da produção e mantendo a qualidade da produção. Os algoritmos Evolucionários Multiobjetivo (AEMOs) são ferramentas eficientes para o tratamento de POMs. Os AEMOs podem ser classificados em três categorias: baseados em dominância de Pareto, baseados em indicadores de qualidade e baseados em decomposição. Embora os *frameworks* baseados em dominância e em indicadores sejam populares, eles têm certas limitações, tais como o tratamento de muitos objetivos para aqueles baseados em dominância e o custo computacional dos *frameworks* baseados em indicadores. Portanto, os AEMOs baseados em decomposição têm atraído a atenção de pesquisadores na última década e são considerados neste trabalho [Trivedi et al. 2017][Zhang et al. 2009]. O MOEA/D (*Multi-Objective Evolutionary Algorithm based on Decomposition*) [Zhang e Li 2007] é um dos mais relevantes algoritmos dessa categoria [Li et al. 2014]. No MOEA/D o problema é decomposto em subproblemas de otimização (mono-objetivo) e cada um deles é tratado individualmente, porém de maneira colaborativa [Zhang e Li 2007, Trivedi et al. 2017]. Na versão utilizada neste trabalho, MOEA/D-DRA [Zhang et al. 2009], o componente DRA (do inglês *Dynamic Resource Allocation*) define a alocação de recursos para cada subproblema com base em um valor de utilidade. Assim, os esforços computacionais do MOEA/D são dirigidos mais frequentemente aos subproblemas com maior utilidade.

As buscas locais têm apresentado bons desempenhos no tratamento de problemas combinatórios [Goldbarg et al. 2015]. Portanto, neste trabalho propõe-se o uso de uma estratégia de busca local atuando no algoritmo MOEA/D-DRA. A estratégia proposta é baseada na heurística NEH[1] [Nawaz et al. 1983] devido aos excelentes resultados desta heurística na solução do PFSP, em especial na versão mono-objetivo do problema [Fernandez-Viagas et al. 2017].

Há diversas abordagens na literatura para solucionar o PFSP multiobjetivo. Algumas são baseadas em Pareto com algoritmo genético [Basseur et al. 2002], outras usam busca local [Minella et al. 2011] e algumas utilizam *Path Relinking* [Zeng et al. 2013]. Como o foco deste trabalho é a heurística baseada em NEH, alguns trabalhos envolvendo uso de NEH para o PFSP merecem destaque. [Li e Li 2015] resolvem o PFSP com uma nova decomposição multiobjetivo baseada na estrutura de busca local

[1] A sigla corresponde às iniciais Nawaz, Enscore e Ham, autores do método [Nawaz et al. 1983]

(MOLSD - *Multiobjective Local Search based Decomposition*). Neste caso, um problema multiobjetivo é decomposto em um número de subproblemas de mono-objetivo usando o método de agregação com otimização simultânea. Os autores utilizam a heurística NEH para inicializar a população para melhorar a qualidade das soluções iniciais. Os dois objetivos utilizados são a minimização do *makespan* e o tempo total de fluxo. Em [Zhe-jian Zhao et al. 2017] é proposto um algoritmo memético, chamado IMOMA (*Improved Multi-objective Memetic Algorithm*), aliado a uma estratégia de inicialização baseada nas heurísticas NEH e LR (iniciais dos autores Liu e Reeves [Zhe-jian Zhao et al. 2017]), além de uma estratégia de busca local. O trabalho utiliza o *makespan* e o tempo total de fluxo como funções objetivo.

Assim como realizado com sucesso em alguns trabalhos da literatura [Li e Li 2015] [Fernandez-Viagas et al. 2017], neste trabalho, a etapa de inicialização da população também é aprimorada como o uso da heurística NEH [Nawaz et al. 1983]. No entanto, a inovação do trabalho vem do uso de uma busca local também inspirada na heurística NEH associada à plataforma do MOEA/D-DRA. Embora uma busca local baseada em Pareto e no operador *insertion* seja associada ao MOEA/D para a solução do PFSP em [Li e Li 2015], a busca local proposta neste trabalho é inspirada em [Chiang et al. 2009]. Este último utiliza a busca local baseada na heurística NEH, porém utilizando um algoritmo memético ao invés do MOEA/D-DRA adotado neste trabalho. Além disso, o que diferencia a busca local de [Chiang et al. 2009] da proposta aqui é a escolha das tarefas a serem removidas e reinseridas em diferentes posições da sequência de tarefas. Em [Chiang et al. 2009] todas as tarefas são reinseridas na ordem que aparecem no subconjunto extraído. Na busca local proposta aqui, as tarefas com menores tempos são selecionadas e a ordem de inserção ocorre das maiores para as menores tarefas. A heurística processa a maior tarefa primeiro e depois realiza o processamento das tarefas menores. Isto ocorre devido ao fato do *makespan* ter um limitante inferior (que é o maior tempo de processamento de uma tarefa). Portanto, assume que a tarefa com maior tempo é a primeira a ser processada, portanto, não sofrendo atraso algum. Assim, as tarefas menores podem ser processadas neste tempo.

O objetivo deste trabalho é explorar (além do método de inicialização) a heurística NEH no mecanismo de busca local que atua ao longo da evolução do MOEA/D-DRA. A abordagem proposta busca definir, para o indivíduo da busca local, um intervalo da solução codificada na qual um percentual de tarefas subsequentes é escolhido para ser avaliado em relação à reinserção, seguindo a heurística NEH. O tamanho do intervalo e o total de tarefas escolhidas são parâmetros do algoritmo. De forma a reduzir o custo computacional da busca local, esta não é realizada de forma completa. Ou seja, após a escolha de uma tarefa para reinserção, nem todas as posições são testadas - apenas algumas escolhidas de forma aleatória. Duas questões de pesquisa levantadas por esse trabalho são (i) a estratégia de definir uma janela seguida pelo subconjunto de elementos a serem reinseridos é uma alternativa viável no mecanismo de busca local? (ii) a abordagem proposta supera o MOEA/D com NEH apenas na inicialização da população? Estas questões deverão ser respondidas com base nos experimentos realizados.

O restante do trabalho está organizado conforme descrito a seguir. A Seção 2 apresenta o problema sendo tratado, o Problema *Flow Shop* de Permutação Multiobjetivo, bem como conceitos relacionados à otimização multiobjetivo e ao algoritmo MOEA/D-

DRA. A Seção 3 descreve a heurística adotada na inicialização da população e inspiradora da estratégia de busca local proposta. O algoritmo proposto é descrito na Seção 4. A Seção 5 contém as simulações realizadas, os resultados e sua análise. As conclusões e direcionamentos futuros são apresentados na Seção 6.

2. Otimização Multiobjetivo e o MO-PFSP

Conforme [Coello et al. 2007], um Problema de Otimização Multiobjetivo (POM) pode ser definido como:

$$\begin{array}{lll} \text{Minimizar:} & \mathbf{f} = \{f_1(\mathbf{x}), \ldots, f_J(\mathbf{x})\} & \\ \text{Sujeito a:} & l_d \leq x_d \leq u_d & d = 1, \ldots, D \end{array} \tag{1}$$

tal que, $\mathbf{f}$ é um vetor com J funções objetivo conflitantes, $\mathbf{x} = (x_1, \ldots, x_D)$ é o vetor de variáveis de decisão, l_d e u_d são os limites inferior e superior, respectivamente, para cada variável de decisão x_d. Variáveis do espaço de decisão que obedecem à restrições definidas em 1 formam o espaço das soluções factíveis e o mapeamento realizado pelas funções objetivo leva ao espaço objetivo. No contexto multiobjetivo, uma solução domina outra se for igual ou melhor que a outra em todos os objetivos e melhor que a outra em pelo menos um objetivo. O conjunto de todas as soluções factíveis que não são dominadas por qualquer outra solução é chamado de conjunto ótimo de Pareto e sua imagem no espaço objetivo é chamada de fronteira de Pareto.

Dentre os diversos algoritmos evolucionários, o MOEA/D (*Multi-Objective Evolutionary Algorithm based on Decomposition*) é um que se destaca com bons resultados na literatura [Trivedi et al. 2017]. O MOEA/D decompõe um POM em um número finito de C subproblemas de otimização escalar, os quais são resolvidos simultaneamente usando alguma função de agregação [Zhang e Li 2007]. Neste trabalho utiliza-se a função de Tchebychef como função de agregação.

O algoritmo MOEA/D original trata todos os subproblemas gerados da mesma maneira, alocando a mesma quantidade de recursos para resolvê-los. Porém, tais subproblemas podem apresentar diferentes complexidades e assim requerem uma quantidade diferente de recursos. O MOEA/D com *Dynamical Resource Allocation* (MOEA/D-DRA) [Zhang et al. 2009] definem e computam um valor utilitário para cada subproblema. Desta forma, os esforços computacionais são atribuídos aos subproblemas com base em seus valores utilitários, quanto maior o valor utilitário associado a um subproblema maior a probabilidade dele ser escolhido.

O Problema *Flow Shop* de Permutação é um problema de alocação de atividades na cadeia de produção de indústrias, em que se deseja programar um conjunto de *N* tarefas a serem executadas por um conjunto de *M* máquinas, e todas as *N* tarefas devem passar pelas máquinas na mesma ordem, ou seja, a primeira tarefa deve ser processada primeiramente na máquina 1, em seguida na máquina 2, e assim sucessivamente até a máquina *M*. Além disso, assume-se que: i) cada tarefa somente é executada por uma máquina a cada instante; ii) cada máquina processa uma tarefa por vez; iii) uma tarefa não pode ser interrompida; iv) todas as tarefas são independentes; v) as máquinas estão sempre disponíveis; e vi) o tempo de preparação está incluso no tempo que leva para processar as tarefas e não depende da sequência de realização das mesmas [Yenisey e Yagmahan 2014]. O solução do PFSP em sua versão multiobjetivo (MO-PFSP) envolve uma sequência de tarefas, isto

é, uma permutação de números $1, \cdots, N$, a serem processadas em cada máquina, de forma a otimizar mais do que uma função objetivo. As duas funções objetivo consideradas neste trabalho são *makespan* e de atraso total (*total tardiness*).

A função objetivo de *makespan* é a mais estudada na literatura para o PFSP e é definida como a seguir. Para uma permutação π, o tempo de conclusão (*completion time*) da tarefa l na última máquina é denotado por $C_l(\pi)$. O *makespan* é definido como o tempo máximo de conclusão $C_{max}(\pi) = \max_l\{C_l(\pi)\}$, ou seja, é o tempo de conclusão da última tarefa na última máquina.

O atraso total utiliza uma data de vencimento para cada tarefa ϕ_l e o o atraso $T_l(\pi)$ da tarefa l é definido por $T_l(\pi) = \max\{0, C_l(\pi) - \phi_l\}$, e o total de atraso é definido por $T(\pi) = \sum_{l=1}^{N} T_l(\pi)$ de acordo com [Yenisey e Yagmahan 2014].

3. Heurística NEH

A NEH (Nawaz-Enscore-Ham) [Nawaz et al. 1983] é uma heurística construtiva eficiente para o PFSP. Ela é baseada na suposição de que uma tarefa com maior tempo de processamento total, considerando todas as máquinas, deve ter prioridade mais alta do que uma tarefa com menor tempo de processamento total.

Existem dois passos principais no algoritmo NEH: 1) ordenar as tarefas; 2) inserir as tarefas uma a uma. O primeiro passo realizado pelo método consiste na ordenação das tarefas de forma decrescente pelo tempo de processamento total. Então, as duas primeiras tarefas são selecionadas (com maiores tempos de processamento total) e a melhor ordem relativa (de acordo com a função objetivo considerada) para elas é identificada, esta ordem não é mais alterada. A seguir a terceira atividade é selecionada (terceiro maior tempo de processamento total) e identifica-se a melhor posição para ela, com relação ao par de tarefas selecionadas no passo anterior, por meio de uma busca exaustiva, ou seja, todas as três possíveis posições são testadas. O próximo elemento do vetor é selecionado e novamente a melhor posição para ele é identificada por meio de uma busca exaustiva. Esta inserção das tarefas na solução é realizada até que a última tarefa, a com menor tempo de processamento total, seja considerada.

Desde a sua criação, foram implementadas diversas variações da NEH [Fernandez-Viagas et al. 2017]. O uso da NEH integrado a heurísticas e metaheurísticas geralmente é feito por meio da inicialização da população. Neste trabalho é implementada uma técnica de busca local inspirada nos conceitos da heurística NEH que atua ao longo da evolução do algoritmo. Diferente da proposta original, a versão proposta neste trabalho evita realizar buscas exaustivas. Isso porque o custo computacional de utilizar a NEH somente na inicialização é relativamente baixo. No entanto, a aplicação de busca local baseada em NEH ao longo de toda a evolução torna-se impraticável para um valor de N elevado.

4. Algoritmo Proposto - MOEA/D-DRA - NEH$_{BL}$

Esta seção traz a descrição da proposta para a solução do *Flow Shop* de Permutação Multiobjetivo. A proposta faz uso da técnica NEH em duas situações distintas: (i) no momento da inicialização da população (ii) na busca local aplicada ao longo da evolução. A seguir, a estratégia de busca global baseada no MOEA/D-DRA, com destaque para a inicialização e o mecanismo de busca local, será descrita.

O Algoritmo 1 apresenta o pseudocódigo da abordagem proposta, que tem por base o algoritmo MOEA/D-DRA incrementado com um mecanismo de inicialização e busca local baseados na heurística NEH. Os primeiros cinco passos do algoritmo correspondem aos procedimentos de inicialização. Os vetores de pesos utilizados para cada subproblema são uniformemente distribuídos e gerados conforme descrito em [Zhang et al. 2009]. A estratégia adotada para a inicialização da população de NP indivíduos busca a melhoria do conjunto inicial de soluções a serem evoluídas pelo MOEA/D-DRA. A heurística NEH é utilizada para gerar 50% das soluções que compõem a população inicial, sendo o restante das soluções gerado de forma aleatória.

Algoritmo 1 Pseudocódigo do MOEA/D-DRA - NEH$_{BL}$

1: Gerar NP vetores de peso $\boldsymbol{\lambda}^i = (\lambda_1^i, \lambda_2^i, ...\lambda_J^i)$, $i = 1, ..., NP$
2: Para $i = 1, \cdots, NP$ definir o conjunto B^i de índices de C vetores de peso próximos a $\boldsymbol{\lambda}^i$
3: Gerar a população inicial $P^0 = \{\mathbf{x}^1, \cdots, \mathbf{x}^{NP}\}$, $\mathbf{x}^i = (x_1^i, x_2^i, ...x_D^i)$ (50% via NEH)
4: Avaliar cada indivíduo em P^0 e associar $\mathbf{x}^i$ com $\boldsymbol{\lambda}^i$
5: Inicializar $\mathbf{z}^* = (z_1^*, \cdots, z_J^*)$ definindo $z_j^* = min_{1 \leq i \leq NP} f_j(\mathbf{x}^i)$, $ger = 1$, $aval = NP$
6: **repeat**
7: **for** cada indivíduo $\mathbf{x}^i$ em NP selecionado com base na utilidade (DRA) **do**
8: **if** $rand < \delta$ **then** *//escopo (rand em U[0,1])*
9: $escopo = B^i$
10: **else**
11: $escopo = \{1, \cdots, NP\}$
12: **end if**
13: Gerar nova solução $\mathbf{y}$ usando o operador genético PMX e a mutação *swap*
14: Avaliar $\mathbf{y}$, $aval = aval + 1$
15: Atualizar $\mathbf{z}^*$, $z_j^* = min(z_j^*, f_j(\mathbf{y}))$,
16: **for** cada subproblema k aleatoriamente selecionado do $escopo$ **do**
17: n_r = 0
18: **if** $n_r < NR$ **then**
19: **if** $g^{tc}(\mathbf{y} \mid \boldsymbol{\lambda}^k, \mathbf{z}^*) < g^{tc}(\mathbf{x}^k \mid \boldsymbol{\lambda}^k, \mathbf{z}^*)$ **then**
20: $\mathbf{x}^k = \mathbf{y}$
21: $n_r = n_r + 1$
22: **else**
23: **if** (random) $p < P_{BL}$ **then**
24: Gerar $\mathbf{y}'$ via BL NEH (Algoritmo 2) aplicada em $\mathbf{y}$
25: **if** $g^{tc}(\mathbf{y}' \mid \boldsymbol{\lambda}^k, \mathbf{z}^*) < g^{tc}(\mathbf{x}^k \mid \boldsymbol{\lambda}^k, \mathbf{z}^*)$ **then**
26: $\mathbf{x}^k = \mathbf{y}'$
27: $n_r = n_r + 1$
28: **end if**
29: **end if**
30: **end if**
31: **end if**
32: **end for**
33: **end for**
34: **if** ger mod $50 == 0$ **then**
35: Atualizar a utilidade de cada subproblema i
36: **end if**
37: $ger = ger + 1$;
38: **until** Critério de parada ser satisfeito

Após essa fase inicial, o algoritmo entra no laço principal (passos 6 a 38). Neste laço, primeiramente é determinado o escopo da iteração atual do processo evolucionário: vizinhança (passo 9) ou toda a população (11). No passo 13 são selecionados aleato-

riamente dois indivíduos de NP e, a partir deles, é gerada uma nova solução $\mathbf{y}$ utilizando operadores genéticos usualmente empregados em permutações (PMX e *swap*). No operador de cruzamento PMX (*Partially-Mapped Crossover*) dois pontos de corte são selecionados aleatoriamente nos indivíduos pais e, em seguida, a sequência de genes delimitada em um dos pais é mapeada para o outro, formando um indivíduo descendente. No operador de mutação *swap*, dois pontos do indivíduo são selecionados aleatoriamente e permutados entre si. Os operadores aplicados geram somente indivíduos factíveis, não sendo necessário procedimento de reparação. Após essas operações, $\mathbf{y}$ é avaliado e há a atualização da solução ideal $\mathbf{z}^*$. No passos 16 a 32 acontece a atualização realizada dentro do escopo. Essa atualização pode se dar de duas formas. Um indivíduo aleatoriamente selecionado dentro do escopo pode ser substituído (i) diretamente pelo indivíduo $\mathbf{y}$ obtido via operadores genéticos (passo 20) (ii) pelo indivíduo $\mathbf{y}'$ modificado por uma busca local baseada em NEH aplicada em $\mathbf{y}$ (passo 26). Em (i), há substituição de $\mathbf{x}^k$ por $\mathbf{y}$ (limitada a Nr substituições) se $\mathbf{y}$ é melhor que $\mathbf{x}^k$ em relação ao k-ésimo subproblema. Caso a nova solução $\mathbf{y}$ seja inferior, o procedimento de busca local é executado no passo 24 com uma probabilidade P_{BL}, pela chamada do Algoritmo 2. Em caso de melhoria após a aplicação da busca local, há substituição. A cada 50 gerações, a variável de utilidade do MOEA/D-DRA é atualizada (passo 35). O processo evolucionário é executado até que o critério de parada seja satisfeito. O algoritmo retorna o conjunto de soluções não-dominadas.

O Algoritmo 2 apresenta o pseudocódigo da busca local proposta.

Algoritmo 2 Busca Local Baseada em NEH

1: **function** BUSCALOCAL(Solução $\mathbf{y}$ do subproblema k)
2: $Intv$= vetor das I tarefas subsequentes obtidas a partir de uma posição aleatória em $\mathbf{y}$
3: $CjobsIns$= vetor das I_{NEH} menores tarefas de $Intv$
4: $\mathbf{y}'$ = $\mathbf{y}$ sem as tarefas de $CjobsIns$
5: Ordenar $CjobsIns$ de forma decrescente em tempo de processamento
6: z = escolha aleatória de um objetivo
7: **for** $i = 1$ a $|CjobsIns|$ **do**
8: Inicializar $f_z(\mathbf{y}'_\mathbf{b}) = \infty$
9: **for** $j = 1$ a $\min\{|\mathbf{y}'|, 50\}$ **do** {50: limite para reduzir o custo da busca}
10: $\mathbf{y}''$ = $\mathbf{y}'$ com a tarefa $CjobsIns[i]$ reinserida aleatoriamente
11: Avaliar $\mathbf{y}''$ (Calcular $f_z(\mathbf{y}'')$)
12: **if** $f_z(\mathbf{y}'') < f_z(\mathbf{y}'_\mathbf{b})$ **then**
13: $\mathbf{y}'_\mathbf{b} = \mathbf{y}''$
14: **end if**
15: **end for**
16: $\mathbf{y}'$ = $\mathbf{y}'_\mathbf{b}$
17: **end for**
18: **return** $\mathbf{y}'$
19: **end function**

Dado um indivíduo $\mathbf{y}$ solução do subproblema k, extrai-se deste indivíduo um subconjunto contíguo de tarefas ($Intv$) de tamanho I a partir de uma posição aleatória de $\mathbf{y}$ (passo 2). Em seguida, um novo subconjunto de tarefas $CjobsIns$ de tamanho I_{NEH} é selecionado de $Intv$ contendo as menores tarefas (de menor duração) de $Intv$ (passo 3). $CjobsIns$ é obtido a partir de um conjunto maior ($Intv$) de tarefas subsequentes obtido de um ponto inicial escolhido aleatoriamente em $\mathbf{y}$. Este mecanismo permite que um número menor de tarefas ($CjobsIns$) seja selecionado para que tarefas de menor duração possam

ser permutadas em $\mathbf{y}$. As tarefas de $CjobsIns$ são então removidas do indivíduo $\mathbf{y}$ (passo 4) para serem reinseridas uma a uma em diferentes posições (passo 10). As reinserções são realizadas da maior para a menor tarefa de $CjobsIns$ e limitado a 50 inserções por tarefa para reduzir o custo computacional da busca em indivíduos de elevada dimensão. A cada reinserção de uma tarefa, a solução parcial resultante $\mathbf{y}''$ é avaliada (passo 11) para o objetivo escolhido no passo 7 e a melhor sequência parcial com esta tarefa é escolhida (passo 13). Ao final da busca local, o melhor indivíduo com todas as tarefas reinseridas é retornado (passo 18).

5. Simulações e Resultados

Esta seção contêm os resultados obtidos com 30 execuções (sementes diferentes) dos algoritmos considerados. O critério de parada é o tempo de processamento em minutos, definido por 10% do número N de tarefas. A Tabela 1 mostra os valores dos parâmetros utilizados. Na busca local, os parâmetros foram fixados levando-se em conta o custo computacional, principalmente para instâncias maiores. Assim, escolhem-se I tarefas (25% do total) para serem permutados. Entretanto, nem todas serão efetivamente permutadas (I_{NEH} é 20% de I), garantindo ainda três tarefas para instâncias menores. A probabilidade P_{BL} de usar a busca local é de 50%. Isso é devido também ao custo computacional da busca local, que pode ser bem intensiva para instâncias maiores. Estes parâmetros foram fixados a partir de ajustes experimentais em testes prévios (não apresentados).

Tabela 1. Parâmetros do algoritmo proposto.

	Valores	**Descrição**
MOEA/D		
NP	100	Tamanho da população.
C	$0,1NP$	Tamanho da vizinhança.
NR	2	Máximo de substituições por uma nova solução.
δ	0,9	Probabilidade de seleção do escopo.
NEH Inicialização		
P	$0,5N$	Percentual da população inicial gerado via NEH.
NEH Busca Local		
I	$0,25N$	Tamanho do subconjunto.
I_{NEH}	min(3 , $0.20I$)	Percentual de tarefas reinseridas do subconjunto.
P_{BL}	0,5	Probabilidade de aplicação da busca local.

Para a execução das simulações são consideradas 11 instâncias do *benchmark* de Taillard, contendo de 20 a 200 tarefas e de 5 a 20 máquinas [Taillard 1993]. Para medir o desempenho dos algoritmos são utilizados dois indicadores de qualidade o Hipervolume e o IGD [Yan et al. 2007]. O Hipervolume é usado para avaliar a convergência e mede o volume do espaço de busca que é dominado pela aproximação da fronteira de Pareto, enquanto o IGD (do inglês *Inverted Generational Distance*) é um critério usado para avaliar também a diversidade e calcula a distância euclidiana entre os pontos da Fronteira de Pareto - soluções exatas ou fronteira de referência - e aqueles obtidos pelo algoritmo de otimização. Neste trabalho a fronteira de referência é composta união das soluções não dominadas obtidas pelos dois algoritmos sendo considerados nas comparações (NEH-MOEAD-DRA e MOEAD-DRA-NEH$_{BL}$).

A análise de significância estatística é feita com base no teste Mann-Whitney-Wilcoxon [Conover 1999] com 95% de confiança. O valor em cinza escuro das tabelas

Tabela 2. Média e desvio padrão para o Hipervolume.

	NEH-MOEAD-DRA	MOEAD-DRA-NEH$_{BL}$
DD_Ta001.txt	$1.38e-01_{1.6e-01}$	$7.51e-01_{0.0e+00}$
DD_Ta011.txt	$4.63e-01_{3.7e-02}$	$5.44e-01_{5.6e-03}$
DD_Ta021.txt	$2.41e-01_{6.8e-02}$	$4.93e-01_{8.0e-02}$
DD_Ta031.txt	$3.43e-01_{1.5e-01}$	$7.68e-01_{8.6e-02}$
DD_Ta041.txt	$1.38e-01_{1.3e-01}$	$4.91e-01_{1.4e-01}$
DD_Ta051.txt	$5.29e-01_{7.2e-02}$	$6.22e-01_{6.5e-02}$
DD_Ta061.txt	$4.46e-01_{1.7e-01}$	$6.65e-01_{1.7e-01}$
DD_Ta071.txt	$4.08e-01_{9.3e-02}$	$5.23e-01_{5.7e-02}$
DD_Ta081.txt	$4.10e-01_{1.1e-01}$	$5.61e-01_{1.0e-01}$
DD_Ta091.txt	$5.26e-01_{1.6e-01}$	$2.74e-01_{1.3e-01}$
DD_Ta101.txt	$4.43e-01_{1.2e-01}$	$3.43e-01_{1.4e-01}$

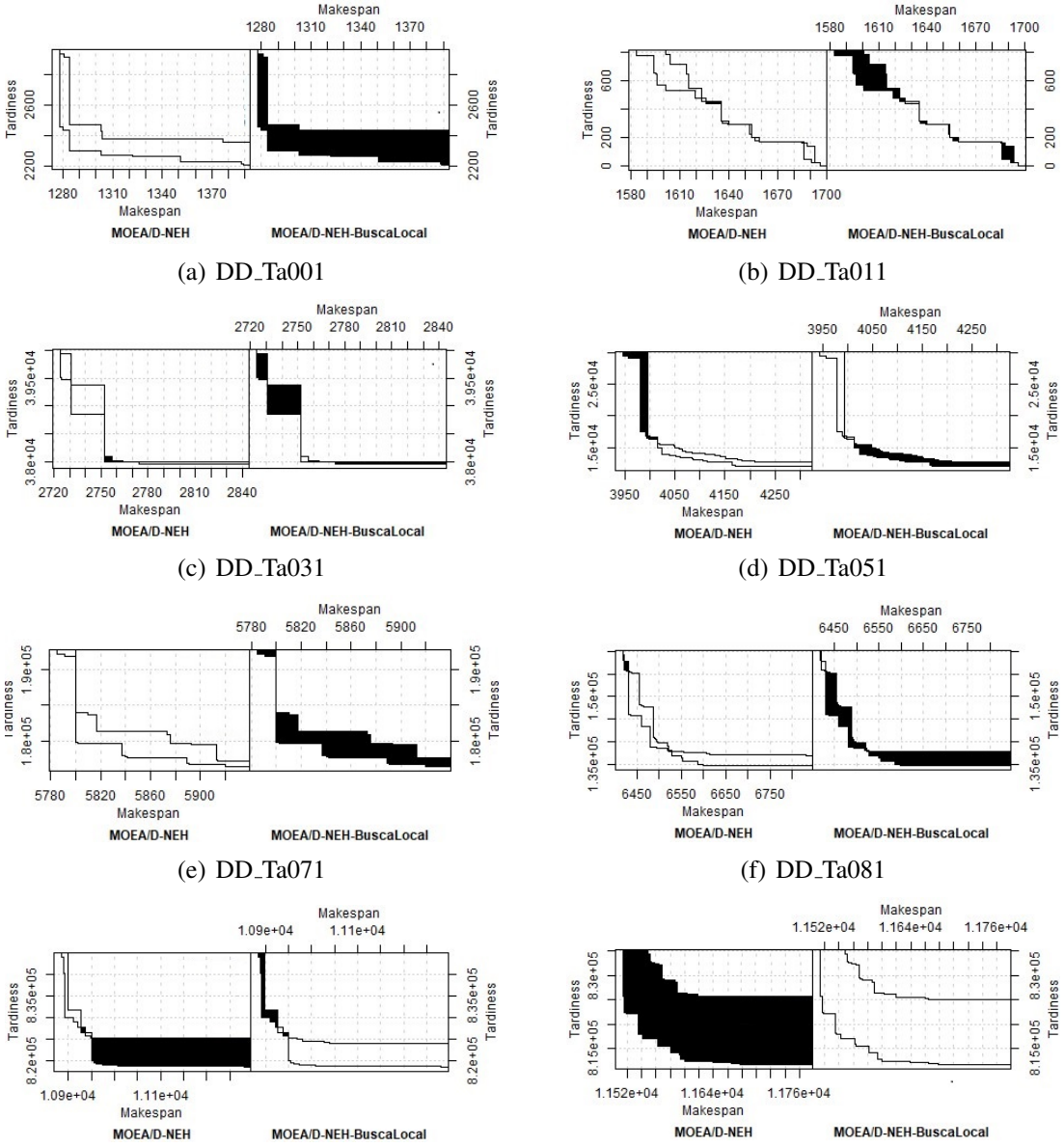
indica o resultado estatisticamente melhor. As Tabelas 2 e 3 mostram que, em nove das 11 instâncias consideradas, a abordagem baseada em busca local que está sendo proposta (MOEAD-DRA-NEH$_{BL}$) é capaz de obter melhores resultados do que o algoritmo que faz uso da NEH somente na inicialização da população (NEH-MOEAD-DRA). De acordo com os valores de Hipervolume e IGD, pode-se considerar que a proposta apresenta melhores resultados tanto em convergência como em cobertura para a maioria (82%) das instâncias consideradas.

Além dos indicadores de qualidade da fronteira de Pareto aproximada, uma análise da função empírica de conquista foi realizada para ilustrar graficamente o comportamento dos algoritmos. Esta análise é apresentada na Figura 1. Os gráficos mostram as funções empíricas associadas aos dois algoritmos, tal que quanto mais escura a cor, maior a probabilidade de atingir os pontos no espaço objetivo. Os eixos são os valores dos objetivos considerados (*makespan* e *total tardiness*).

As funções empíricas da Figura 1 mostram que o algoritmo MOEA/D-DRA-NEH$_{BL}$ obteve uma melhor aproximação da fronteira de Pareto em comparação ao algoritmo proposto NEH-MOEA/D-DRA considerando *makespan* (Figura 1 (a), (b), (c) e (f)) e *total tardiness* (Figura 1 (a), (b), (c) e (f)) para a maioria das instâncias com até 100 tarefas. Porém, para as instâncias com 200 tarefas (Figura 1 (g) e (h)) o algoritmo proposto é superado pelo algoritmo NEH-MOEA/D-DRA segundo as funções empíricas. Para instâncias maiores (número maior de tarefas) são consideradas, o MOEA/D-DRA-NEH$_{BL}$ perde diversidade, o que poderia explicar o baixo desempenho para estas instâncias.

6. Conclusões e Trabalhos Futuros

No presente trabalho foi apresentada uma proposta de busca local para ser utilizada em uma plataforma multiobjetivo para a solução do *Flow Shop* de Permutação. A abordagem multiobjetivo envolveu a minimização do *makespan* e do tempo total de atraso e teve por base o algoritmo MOEA/D-DRA. Neste trabalho foi proposto um mecanismo de busca local baseado na heurística NEH a ser incorporado ao MOEA/D-DRA. O algoritmo proposto foi comparado com a versão do MOEA/D-DRA utilizando NEH apenas na inicialização da população. Ambas as abordagens foram aplicadas na solução de 11 instâncias do *Flow Shop* de permutação com tamanhos variando de 20 a 200 tarefas e 5 a 20 máquinas. Os resultados mostraram que a abordagem proposta é extremamente



(a) DD_Ta001 (b) DD_Ta011

(c) DD_Ta031 (d) DD_Ta051

(e) DD_Ta071 (f) DD_Ta081

(g) DD_Ta091 (h) DD_Ta101

Figura 1. Funções empíricas de conquista do MOEA/D-NEH $\times$ MOEA/D-NEH$_{BL}$ considerando (a) 20 tarefas e 5 máquinas, (b) 20 tarefas e 10 máquinas, (c) 50 tarefas e 5 máquinas, (d) 50 tarefas e 20 máquinas, (e) 100 tarefas e 10 máquinas, (f) 100 tarefas e 20 máquinas, (g) 200 tarefas e 10 máquinas e (h) 200 tarefas e 20 máquinas.

Tabela 3. Média e desvio padrão para o IGD.

	NEH-MOEAD-DRA	MOEAD-DRA-NEH$_{BL}$
DD_Ta001.txt	$4.41e-02_{4.0e-02}$	$6.92e-05_{9.1e-05}$
DD_Ta011.txt	$2.49e-03_{7.4e-04}$	$4.73e-04_{8.7e-05}$
DD_Ta021.txt	$6.95e-03_{1.3e-03}$	$2.84e-03_{1.9e-03}$
DD_Ta031.txt	$6.98e-02_{1.3e-01}$	$1.80e-02_{5.5e-03}$
DD_Ta041.txt	$4.41e-02_{1.5e-02}$	$1.73e-02_{7.3e-03}$
DD_Ta051.txt	$1.64e-02_{4.2e-03}$	$1.14e-02_{3.2e-03}$
DD_Ta061.txt	$4.03e-02_{1.5e-02}$	$2.38e-02_{1.2e-02}$
DD_Ta071.txt	$2.01e-02_{5.8e-03}$	$1.35e-02_{2.9e-03}$
DD_Ta081.txt	$2.42e-02_{7.6e-03}$	$1.58e-02_{5.1e-03}$
DD_Ta091.txt	$2.42e-02_{9.6e-03}$	$4.17e-02_{1.0e-02}$
DD_Ta101.txt	$2.63e-02_{7.3e-03}$	$3.29e-02_{9.9e-03}$

competitiva uma vez que obteve melhores resultados em 9 das 11 instâncias consideradas. Considerando as questões de pesquisa levantadas na introdução, pode-se avaliar que a estratégia de definir uma janela, seguida pelo subconjunto de elementos a serem reinseridos, é uma alternativa viável para se implementar uma busca local. A versão proposta foi capaz de superar a versão do MOEA/D que implementa somente a NEH na fase de inicialização da população para a maioria (82%) das instâncias consideradas. A busca local proposta procurou balancear esforço computacional com exploração do espaço de busca. Porém, isso é fortemente dependente da escolha dos parâmetros da busca. Como trabalhos futuros, é possível considerar técnicas que avaliem este compromisso. Uma alternativa atraente é o uso de hiper-heurística para determinar os parâmetros, inclusive ao longo da evolução.

Agradecimentos

Os autores agradecem às agências de fomento pelo suporte financeiro parcial à pesquisa.

Referências

Baker, K. R. e Trietsh, D. (2009). *Principles of Sequencing and Scheduling*, volume 1. Wiley.

Basseur, M., Seynhaeve, F., e Talbi, E. (2002). Design of multi-objective evolutionary algorithms: application to the flow-shop scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1151–1156 vol.2.

Chiang, T.-C., Cheng, H.-C., e Fu, L.-C. (2009). Multiobjective permutation flow shop scheduling using a memetic algorithm with an NEH-based local search. In *LNCS*, volume 5754, pages 813–825.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.

Conover, W. J. (1999). *Practical Nonparametric Statistics*. Wiley, 3 edition.

Fernandez-Viagas, V., Ruiz, R., e Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European J. of Operational Research*, 257(3):707–721.

Garey, M. R., Johnson, D. S., e Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129.

Goldbarg, M. C., F., G. E., e Luna, H. P. (2015). *Otimização Combinatória e Metaheurísticas- Algoritmos e Aplicações*, volume 1. Campus.

Li, K., Fialho, A., Kwong, S., e Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130.

Li, X. e Li, M. (2015). Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. *IEEE Transactions on Engineering Management*, 62:544–557.

Minella, G., Ruiz, R., e Ciavotta, M. (2011). Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11):1521 – 1533.

Nawaz, M., Enscore Jr, E. E., e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega - Elsevier*, 11(1):91–95.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285.

Trivedi, A., Srinivasan, D., Sanyal, K., e Ghosh, A. (2017). A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462.

Yan, J., Li, C., Wang, Z., Deng, L., e Sun, D. (2007). Diversity metrics in multi-objective optimization: Review and perspective. In *2007 IEEE International Conference on Integration Technology*, pages 553–557.

Yenisey, M. M. e Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 45:119–135.

Zeng, R.-Q., Basseur, M., e Hao, J.-K. (2013). Solving bi-objective flow shop problem with hybrid path relinking algorithm. *Applied Soft Computing*, 13(10):4118 – 4132.

Zhang, Q. e Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

Zhang, Q., Liu, W., e Li, H. (2009). The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 203–208. IEEE.

Zhe-jian Zhao, Xue-qing He, e Feng Liu (2017). An improved multi-objective memetic algorithm for bi-objective permutation flow shop scheduling. In *2017 International Conference on Service Systems and Service Management*, pages 1–6.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., e Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49.