PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

ESCOLA POLITÉCNICA

PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS

MÁRCIA DE FÁTIMA MORAIS

ABORDAGENS DE EVOLUÇÃO DIFERENCIAL PARA OTIMIZAÇÃO EM PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO EM AMBIENTES *FLOW SHOP* PERMUTACIONAL

CURITIBA

2017

MÁRCIA DE FÁTIMA MORAIS

ABORDAGENS DE EVOLUÇÃO DIFERENCIAL PARA OTIMIZAÇÃO EM PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO EM AMBIENTES *FLOW SHOP* PERMUTACIONAL

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutora em Engenharia de Produção e Sistemas.

Orientador: Prof. Dr. Leandro dos Santos Coelho

CURITIBA

2017

Morais, Márcia de Fátima
M827a 2017 Abordagens de evolução diferencial para otimização em problemas de programação da produção em ambientes *Flow Shop* permutacional / Márcia de Fátima Morais ; orientador: Leandro dos Santos Coelho. – 2017.
196 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2017
Bibliografia: f. 152-169

1. Engenharia de produção. 2. Programação (Computadores). 3 Algoritmos. I. Coelho, Leandro dos Santos. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Engenharia de Produção e Sistemas. III. Título

CDD 22. ed. – 670

MÁRCIA DE FÁTIMA MORAIS

ABORDAGENS DE EVOLUÇÃO DIFERENCIAL PARA OTIMIZAÇÃO EM PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO EM AMBIENTES *FLOW SHOP* PERMUTACIONAL

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutora em Engenharia de Produção e Sistemas.

COMISSÃO EXAMINADORA

Dr. Leandro dos Santos Coelho
Pontifícia Universidade Católica do Paraná (PUCPR)

Maria Teresinha Arns Steiner
Pontifícia Universidade Católica do Paraná (PUCPR)

Dr. Cassius Tadeu Scarpin
Universidade Federal do Paraná (UFPR)

Dr. Gustavo Valentim Loch
Universidade Federal do Paraná (UFPR)

Dra. Viviana Cocco Mariani
Pontifícia Universidade Católica do Paraná (PUCPR)

Curitiba, 6 de Dezembro de 2017.

Dedico este trabalho

À minha mãe Divanir e à minha filha Sofia, as maiores bençãos da minha vida.

AGRADECIMENTOS

Agradeço a Deus, por abençoar cada dia da minha vida, por iluminar meu caminho e me dar forças para seguir sempre em frente.

Agradeço a minha Mãe Divanir, pelo apoio incondicional e pela compreensão nos dias em que me fiz ausente e pelos incentivos nos dias de “transpiração”.

Agradeço ao meu Pai Acioly (*in memorian*), cujos valores morais, sociais, éticos e religiosos que hoje me definem são suas maiores heranças. Um exemplo de vida que sempre busco seguir.

Minha eterna gratidão ao meu orientador, Professor Leandro, pela oportunidade a mim concedida de poder desfrutar de toda a sua sabedoria e competência, por seus inestimáveis conselhos, tanto profissionais quanto pessoais e pela confiança no meu potencial acadêmico,.

Agradeço aos meus queridos Tios, Divonzir e Sônia pela calorosa acolhida em Curitiba, que tornaram minha jornada de viagens e estudos, mais leve e divertida no decorrer dos anos de Doutorado.

Agradeço aos meus grandes Amigos e Compadres, Claudilaine e Marcos, que estão sempre ao meu lado, me incentivando e me apoiando das mais diversas formas.

Agradeço a PUCPR (Pontifícia Universidade Católica do Paraná) pelo apoio imprescindível por meio da isenção de taxa, sem a qual não teria sido possível a realização do Doutorado.

Agradeço a todos os funcionários e professores do PPGEPS da PUCPR, em especial a Denise, pela constante atenção e suporte acadêmico, principalmente nos momentos em que estive distante em função de minhas atividades profissionais.

Agradeço a Universidade Estadual do Paraná, Campus de Campo Mourão, em especial a cada um dos meus colegas de trabalho do Colegiado de Engenharia de Produção, pelo apoio durante o período de realização do Doutorado.

Aos membros da banca examinadora, pelas valiosas contribuições feitas a esta tese, meus respeitosos agradecimentos.

A todos que de uma maneira direta e indireta contribuíram para o desenvolvimento deste estudo, muito obrigada.

“A nobreza do homem está em sentir gratidão pelas graças recebidas e gravá-las no seu coração”.

(Meishu Sama)

RESUMO

A Evolução Diferencial (ED) é uma metaheurística estocástica da Computação Evolutiva, originalmente desenvolvida para solução de problemas complexos de otimização irrestrita em domínio contínuo. Nos últimos anos a ED vem obtendo destaque no cenário científico e tem se mostrado apropriada também para resolver problemas de otimização combinatória. Nesta tese, cujo objetivo principal consiste no desenvolvimento de novos métodos de solução baseados em abordagens de ED, foi considerado o Problema de Programação da Produção (PPP) em ambientes *Flow Shop* Permutacional (FSP), um típico problema de otimização combinatória que tem sido amplamente estudado devido a sua significância prática e acadêmica. Embora a literatura especializada contemple diversas aplicações da abordagem de ED ao PPPs, uma revisão de trabalhos orientados à solução de PPPs em FSP utilizando abordagens de ED, mostrou que poucos estudos adotam abordagens de ED autoadaptativas para a solução do problema em questão. Diante do exposto, como principal contribuição desta tese, destaca-se o desenvolvimento de uma variante do algoritmo de ED Discreta e duas novas variantes do algoritmo de ED Autoadaptativo para a solução de PPPs em FSP, um problema combinatório de elevada complexidade. Nos algoritmos desenvolvidos, foram combinadas, a versão clássica do algoritmo de ED, a regra *Largest-Order-Value* para conversão dos domínios, uma nova estratégia de ED, bem como mecanismos autoadaptativos para seleção da estratégia de ED e configuração dos parâmetros de controle. Os algoritmos desenvolvidos, denominados EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 foram executados em 110 casos testes (*Benchmarks*) clássicos da literatura. O desempenho dos algoritmos foi avaliado em termos da qualidade das soluções fornecidas, por meio dos erros relativos percentuais em relação aos limitantes superiores (*Makespan*) da função objetivo disponíveis na literatura especializada, e em termos da capacidade de exploração do espaço de busca por meio do desvio padrão e da convergência. Os algoritmos também foram avaliados em termos de tempos médios computacionais. Para os casos testes de Taillard, o algoritmo EDDA-FSP_1 apresentou melhor desempenho em termos de erros relativos percentuais e desvio padrão. Para os casos testes de Carlier e de Heller os algoritmos EDDA-FSP_1 e EDDA-FSP_2 apresentaram o mesmo desempenho em termos de erros relativos percentuais, enquanto que o algoritmo EDDA-FSP_2 apresentou melhor desempenho em termos de desvio padrão. Para os casos testes de Reeves o algoritmo EDDA-FSP_2 apresentou melhor desempenho em termos de erros relativos percentuais, enquanto que o algoritmo EDDA-FSP_1 apresentou melhor desempenho em termos de desvio padrão. Na análise de convergência, para os casos selecionados (10 de 110), o algoritmo EDDA-FSP_1 apresentou melhor desempenho em termos de convergência em 30% dos casos, o algoritmo EDDA-FSP_2 apresentou melhor desempenho para outros 30% dos casos, enquanto que os algoritmos EDDA-FSP_1 e EDDA-FSP_2 apresentaram desempenho semelhante em 40% dos casos. Em relação aos tempos médios computacionais, o algoritmo EDDA-FSP_2 apresentou os menores tempos computacionais para a maioria dos casos dos *benchmarks* de Carlier, Heller e Reeves, enquanto que o algoritmo EDDA-FSP_1 apresentou os menores tempos computacionais para a maioria dos casos do *benchmark* de Taillard. De forma geral os resultados mostram que as abordagens propostas nesta tese são promissoras para a solução de PPPs em ambientes FSP.

Palavras-Chave: Métodos de Solução. *Flow Shop* Permutacional. *Makespan*. Evolução Diferencial Discreta. Evolução Diferencial Autoadaptativa.

ABSTRACT

Differential Evolution (DE) is a stochastic metaheuristic approach of Evolutionary Computing originally developed to solve complex problems of unconstrained optimization in the continuous domain. In recent years, DE has achieved prominence in the scientific scenario and has also proved suitable for solving combinatorial optimization problems. In this thesis, whose main objective was the development of new solution methods based on DE approaches, we considered the Scheduling Problems (SP) in Permutational Flow Shop (PFS), a typical combinatorial optimization problem that has been widely studied because of its practical and academic significance. Although the literature considers several applications of the DE approach to SP, a review of studies oriented to the solution of SP in PFS using DE approaches showed that few studies adopt Self-Adaptive DE methods to solve the problem in question. In this regard, the main contribution of this thesis is the development of a variant of the Discrete DE algorithm and two new variants of the Self-Adaptive DE algorithm for the solution of SP in PFS, a combinatorial problem of high complexity. In the developed algorithms, the classical version of the DE algorithm, the largest-order-value rule for domain conversion, a new DE strategy as well as self-adaptive mechanisms for selection of the DE strategy and configuration of its control parameters were combined. The developed algorithms, named EDD-FSP, EDDA-FSP_1 and EDDA-FSP_2, were executed in 110 classic *Benchmark*s of the literature. The performance of the algorithms was evaluated in terms of the quality of solutions provided, by means of the percentage relative errors in relation to the upper bounds (Makespan) of the objective function available in the specialized literature, and in terms of the search-space exploration capacity through standard deviation and convergence information. The algorithms were also evaluated in terms of average computational times. For the cases of the Taillard *Benchmark*, the EDDA-FSP_1 algorithm presented better performance in terms of percentage errors and standard deviation. For the cases of the Carlier and Heller *Benchmark*s, the EDDA-FSP_1 and EDDA-FSP_2 algorithms showed the same performance in terms of percentage errors, whereas the EDDA-FSP_2 algorithm displayed better performance in terms of standard deviation. For the cases of the Reeves *Benchmark*, the EDDA-FSP_2 algorithm presented better performance in terms of percentage errors, while the EDDA-FSP_1 algorithm showed better performance in terms of standard deviation. In the convergence analysis, for the selected cases (10 of 110), the EDDA-FSP_1 algorithm presented better performance in 30% of the cases; the EDDA-FSP_2 algorithm exhibited better performance for another 30% of the cases; and the EDDA-FSP_1 and EDDA-FSP_2 algorithms showed similar performance for 40% of the cases. The EDDA-FSP_2 algorithm presented the lowest computational times for most cases of the Carlier, Heller and Reeves benchmarks, while the EDDA-FSP_1 algorithm presented the lowest computational times for most cases of the benchmark from Taillard. Overall the results show that the approaches proposed in this thesis are promising for the PPP solution in FSP environments.

Key-words: Solution Methods. Permutational Flow Shop. Makespan. Discrete Differential Evolution. Self-Adaptive Differential Evolution.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

ACO	*Ant Colony Optimization*
AE	Algoritmo Evolutivo
CE	Computação Evolucionária
BT	Busca Tabu
CE	Custos de Estoque
CoDE	*Composite Differential Evolution*
CT	Custos de Transporte
CR	Fator ou taxa de cruzamento
DABC	*Discrete Artificial Bee Colony*
DEahcSPX	*Differential Evolution using Adaptive Hill-Climbing Crossover-based Local Search and Simplex Crossover*
DEGL	*Differential Evolution with Global and Local neighborhoods*
DHS	*Discrete Harmonic Search*
ED	Evolução Diferencial
EDA	Evolução Diferencial Adaptativa
EDE	*Enhanced Differential Evolution*
EDD	*Earliest Due Date*
EDD	Evolução Diferencial Discreta
DDE	*Discrete Differential Evolution*
EPSDE	*Ensemble of Parameters and mutation Strategies in Differential*
F	Fator ou taxa de mutação
FF	*First-Fit*
FSP	*Flow Shop* Permutacional
FST	*Flow Shop* Tradicional
GA	*Genetic Algorithm*
GRASP	*Greedy Randomized Adaptive Search Procedure*
HDDE	*Hybrid Discrete Differential Evolution*
HGA	*Hybrid Genetic Algorithm*
HT	Teste de Hipótese
IC	Inteligência Computacional
IEEE	*Institute of Electrical and Electronics Engineers*

IG	*Iterated Greedy*
IGA	*Improved Genetic Algorithm*
ILS	*Iterated Local Search*
JADE	*J Adaptive Differential Evolution*
jDE	*j Differential Evolution*
LIBT	*Least Idle/Blocking Time*
LOV	*Largest Order Value*
LPSR	*Linear Population Size Reduction*
LPV	*Largest Position/Parameter Value*
LRV	*Largest- Ranked-Value*
LRVA	*Largest Roating Angle Value*
L-SHADE	*Success-History Based Adaptive Differential Evolution With Linear Population Size Reduction*
LSL	*Smallest Slack Time on the Last Machine*
MAEA	*Mutiagent Evolutionary Algorithm*
MOEA	*Multiagent Evolutionary Algorithm*
MBT	Tempo de Bloqueio das Máquinas
MIT	Tempo Ocioso Total
NEH	Nawaz, Enscore e Ham
NFE	Número Máximo de Avaliações da Função Objetivo
NP-*Hard*	*Non-Polinomial Hard*
NT	Número de tarefas em atraso
OCBA	*Optimal Computing Budg*et al*location*
OGA	*Othogonal Genetic Algorithm*
OSL	*Smallest Overall Slack Time*
PP	Programação da Produção
PPCP	Planejamento, Programação e Controle da Produção
PPP	Problema de Programação da Produção
PSO	*Particle Swarm Optimization*
QDEA	*Quantum Differential Evolutionary*
QEA	*Quantum-Inspired Evolutionary*
RDPT	Tempos de Processamento Dependentes de Recursos
RPI	*Relative Position Indexing*

RLS	*Referenced Local Search*
RPI	Índice de Posição Relativa
SA	*Simulated Annealing*
SaDE	*Self-adaptive Differential Evolution*
SHADE	*Success-History based Adaptive Differential Evolution*
SOMA	*Self Organizing Migrating Algorithm*
SPS-L-SHADE-EIG	*Success-History Based Adaptive Differential Evolution With Linear Population Size Reduction with an Eigenvector-Based Crossover and a Successful-Parent Selecting*
SPV	*Smallest Position/ Parameter Value*
SPVF	*Smallest Parameter Value First*
TCO	Custo Total de Oportunidade
TCU	Custo Total de Utilidade
TL	*Tabu List*
TSC	Custo Total de *Setup*
TST	Tempo Total de *Setup*
VIG	*Variable Iterated Greed*
VNS	*Variable Neighborhood Search*
WIP	*Work-In-Process*

LISTA DE SÍMBOLOS

A	Arquivo externo formado pelo conjunto de soluções inferiores recentemente exploradas e armazenadas utilizadas pela estratégia *DE/target-to-pbest/1* com arquivo
$A^{(SPS)}$	Arquivo do número de soluções recentemente atualizadas
b_L	Limitante inferior
b_U	Limitante superior
C_j	*Completion Time* ou data de término da *j*-ésima tarefa
C_{max}	*Makespan* ou Duração Total da Programação
CR	Fator de cruzamento
CR_i	Taxa de cruzamento associado ao *i*-ésimo indivíduo da população atual
CR_{m_k}	Média da distribuição normal associada a k-ésima estratégia utilizada para gerar a taxa de cruzamento
D	Número de parâmetros do problema a ser otimizado
d_j	Data de entrega da *j*-ésima tarefa
E_j	*Earliness* ou Adiantamento da *j*-ésima tarefa
F	Fator de mutação
F_i	Fator de mutação associado ao *i*-ésimo indivíduo da população atual
F_j	*Flow Time* ou Tempo de fluxo da *j*-ésima tarefa
F_l	Limite inferior do fator de mutação
F_u	Limite superior do fator de mutação
$f(x)$	Função objetivo ou função a ser otimizada
G	Geração atual do processo de otimização
$g_{i,G}$	Vetor doador global relativo ao i-ésimo vetor alvo da geração G
G_{max}	Número máximo de gerações do processo de otimização
J_j	*j*-ésimo elemento do conjunto de tarefas a serem processadas
J_n	*n*-ésimo elemento do conjunto de tarefas a serem processadas
j_{rand}	Índice escolhido aleatoriamente
$L_{i,G}$	Vetor doador local relativo ao i-ésimo vetor alvo da geração G
l_j	Data de liberação da *j*-ésima tarefa

L_j	*Lateness* ou Atraso da *j*-ésima tarefa
L_{max}	Adiantamento (*Latenss*) Máximo
LP	Número de gerações necessárias para atualizar as probabilidades $p_{k,G}$ (período de aprendizagem)
m	Conjunto de máquinas disponíveis para processamento
M_{CR}	Arquivo responsável por armazenar o histórico do parâmetro *CR*
M_F	Arquivo responsável por armazenar o histórico do parâmetro *F*
M_k	*k*-ésimo elemento do conjunto de máquinas disponíveis para processamento
M_m	*m*-ésimo elemento do conjunto de máquinas disponíveis para processamento
n	Conjunto de tarefas a serem processadas
N^{init}	Tamanho da população na geração 1
N^{min}	Tamanho da população na geração final
N_p	Tamanho da população
$ns_{k,G}$	Número de vetores experimentais bem sucedidos gerados na geração *G* pela k-ésima estratégia
$ns_{f,G}$	Número de vetores experimentais gerados pela k-ésima estratégia e descartados durante o processo de seleção na geração *G*
op_{kj}	Operação da *j*-ésima tarefa na *k*-ésima máquina
$p_{k,G}$	Probabilidade da k-ésima estratégia ser escolhida na geração *G*
P_x	População corrente
P_u	População mutante
$q_{i,G}$	Número de atualizações consecutivas recentemente malsucedidas da *i*-ésima solução na *G*-ésima geração
q_{ijk}	Tempo de conclusão da *j*-ésima tarefa na *k*-ésima posição na *i*-ésima máquina
$rand[0,1]$	Gerador de números aleatórios com distribuição uniforme no intervalo [0,1]
$randc_i$	Valor selecionado aleatoriamente da distribuição de Cauchy
$randn_i$	Valor selecionado aleatoriamente da distribuição Normal

$rand_{i,j}[0,1]$	Gerador de números aleatórios com distribuição uniforme no intervalo [0,1] avaliado independentemente para cada j-ésima componente do i-ésimo indivíduo
S_{CR}	Conjunto de todas as taxas de cruzamento bem sucedidas na geração *G*
S_F	Conjunto de todos os fatores de mutação bem sucedidos na geração *G*
t_{ij}	Tempo de processamento da *j*-ésima tarefas na *i*-ésima máquina
T_j	*Tardiness* ou Atraso da *j*-ésima tarefa
T_{max}	Atraso Máximo
$u_{i,G}$	Vetor experimental relativo ao *i*-ésimo vetor alvo da geração *G*
$\hat{u}_{i,G}$	Resultado do mapeamento do vetor obtido após as operações de Mutação e cruzamento para um vetor de números inteiros
U_j	Unidade de penalização da *j*-ésima tarefa
$u_{j,i,G}$	j-ésima componente do vetor experimental $u_{i,G}$
$v_{i,G}$	Vetor doador relativo ao *i*-ésimo vetor alvo da geração *G*
$v_{j,i,G}$	j-ésima componente do vetor doador $v_{i,G}$
x^*	Ótimo da função ou ótimo global
$x_{best,G}$	Indivíduo da população na geração *G* que apresenta o menor valor de função objetivo
$x_{i,G}$	i-ésimo indivíduo da população na geração *G*
$\hat{x}_{i,G}$	Resultado do mapeamento do vetor de permutação de inteiros $x_{i,G}$ em um vetor de números reais
$x_{i,0}$	Indivíduo pertencente à população inicial
X_{jk}	Variável binária, que assume 1 se a *j*-ésima tarefa na *k*-ésima posição ou 0 em caso contrário
x_j^{mim}	Valor mínino para o elemento do vetor mutante na *j*-ésima dimensão
$x_j^{\max}$	Valor máximo para o elemento do vetor mutante na *j*-ésima dimensão
x^L	Limitante inferior imposto as variáveis
x^U	Limitante superior imposto as variáveis
w	Fator de peso para combinar os modelos de vizinhança local e global

Ω	Região viável ou espaço de busca
$\Re^n$	Conjunto das *n*-duplas ordenadas de números reais
Δ_{ij}	Matriz de diferença
τ_1	Probabilidade de alteração dos parâmetros F_i
τ_2	Probabilidade de alteração dos parâmetros CR_i
φ	Tamanho do arquivo externo *A*
η	Fator de escala utilizado para a geração do vetor $L_{i,G}$ e $g_{i,G}$
ψ	Fator de escala utilizado para a geração do vetor $L_{i,G}$ e $g_{i,G}$
μ_{CR}	Média da distribuição normal utilizada para gerar a taxa de cruzamento
μ_F	Parâmetro de locação da distribuição de Cauchy utilizada para gerar o fator de mutação
$\sum C_j$	Tempo Total de Conclusão
$\sum Sj$	Tempo Total de *Setup*
ΣU_j	Número de Tarefas em Atraso
ΣW_j	Tempo Total de Espera
$\sum w_j C_j$	Tempo Total de Conclusão Ponderado
$\Sigma w_j F_j$	Tempo de Fluxo Total Ponderado
$\Sigma w_j L_j$	Adiantamento Total Ponderado
$\Sigma W_j/n$	Tempo Médio de Espera
$\Sigma w_j T_j$	Atraso Total Ponderado
$\Sigma w_j W_j$	Tempo Total de Espera Ponderado

SUMÁRIO

Capítulo 1 - INTRODUÇÃO

A otimização é uma área do conhecimento geralmente vinculada a Matemática que visa minimizar ou maximizar pelo menos uma função objetivo (PARK; JEONG; LEE, 2006). Embora seja notadamente reconhecido que a otimização está em muitas aplicações, abrangendo Engenharias, Computação, Finanças, entre outras áreas do conhecimento, isto não torna a resolução de problemas de otimização menos complexa (YANG, 2010; BOUSSAÏD; LEPAGNOT; SIARRY, 2013; MAHDAVI; SHIRI; RAHNAMAYAN, 2015; WARI; ZHU, 2016).

Nos últimos 30 anos, diversos novos algoritmos de otimização foram desenvolvidos e aplicados a diversos problemas de otimização (SAVSANI; SAVSANI, 2016). No entanto, conforme sugere o Teorema *No Free Lunch* (WOLPERT; MACREADY, 1997) nenhum algoritmo é adequado para todos os problemas. Assim, mais pesquisa é necessária para investigar diferentes algoritmos de otimização, modificar algoritmos existentes e desenvolver novos algoritmos para atender às aplicações da vida real com alta eficiência (WANG; CAI; ZHANG, 2011, 2012; WANG et al., 2014).

Neste contexto, enquadram-se as pesquisas na área da Computação Evolutiva (CE), um ramo da pesquisa emergente da Inteligência Computacional (IC) ou de uma área mais abrangente denominada Computação Natural (CN). A CE propõem um novo paradigma para solução de problemas de busca e otimização, que se preocupa com a investigação de sistemas inspirados pela teoria neoDarwiniana da evolução por meio da seleção natural (BÄCK et al*.,* 2000; BROWNLEE, 2011; SIMON, 2013; EIBEN; SMITH, 2015).

A CE compreende um conjunto de algoritmos, denominados Algoritmos Evolutivos ou Evolucionários (AEs), que associa os princípios da teoria da evolução e da seleção natural para a resolução de problemas de otimização complexos (GOLDBERG, 1989; BÄCK et al*.,* 2000; BRANKE, 2002; DAS; SUGANTHAN, 2011; EIBEN; SMITH, 2015).

A ideia subjacente comum geralmente adotada em AEs é a de que dada uma população de indivíduos dentro de algum ambiente com recursos limitados, a competição por esses recursos faz com que ocorra a seleção natural ou a sobrevivência do mais apto (EIBEN; SMITH, 2015). Todos os AEs tem em comum o

conceito de simulação da evolução das espécies por meio da seleção, mutação e reprodução, processos esses que dependem do desempenho dos indivíduos destas espécies dentro do ambiente (LINDEN, 2012).

Os AEs bem conhecidos por sua capacidade de lidar com problemas de otimização não lineares e complexos (MALLIPEDDI et al., 2011), têm sido uma escolha atrativa para solução dos mais variados tipos de problemas. Isto se deve ao desempenho robusto e confiável, capacidade de busca global, pouca ou nenhuma exigência de informação sobre o problema, bem como a facilidade de implementação, o paralelismo e a não exigência de uma função objetivo diferenciável ou contínua (BÄCK et al., 2000; NORMAM; IBA, 2008; MALLIPEDDI et al., 2011).

Segundo Pierezan et al. (2017) os AEs permitem que os projetistas abordem os problemas de otimização tentando iterativamente melhorar uma população de soluções candidatas em relação a uma determinada medida de qualidade. Embora não garantam uma convergência, os AEs são capazes de detectar soluções de qualidade que possam ser de interesse para os usuários.

Os AEs segundo Bäck et al. (2000), Feoktistov (2006), Brownlee (2011), Lopes e Takahashi (2011), Binitha e Sathya (2012), Simon (2013) e Eiben e Smith (2015) tradicionalmente incluem Algoritmos Genéticos (HOLLAND, 1962; HOLLAND, 1975), Programação Evolutiva (FOGEL, 1962; FOGEL, 1964), Estratégias Evolutivas (RECHENBERG, 1965; SCHWEFEL, 1965) e Programação Genética (KOZA, 1989; KOZA, 1992).

Além destes, o algoritmo de Evolução Diferencial (ED), também categorizado como um AE, desenvolvido por Storn e Price em 1995, vem obtendo destaque no cenário científico nos últimos anos, devido a seus conceitos simples, sua facilidade de implementação, rapidez na convergência, eficiência na busca de soluções adequadas para problemas complexos de otimização e facilidade de implementação em qualquer linguagem de programação padrão (MALLIPEDDI et al., 2011; DAS; SUGANTHAN, 2011; SUGANTHAN, 2012; TANABE; FUKUNAGA, 2014; DAS; MULLICK; SUGANTHAN, 2016). Além disso, o algoritmo de ED é considerado robusto e capaz de fornecer os mesmos resultados de forma consistente em várias simulações (PIEREZAN et al., 2017).

O algoritmo de ED, assim como outros AEs, é um otimizador global estocástico populacional que utiliza operadores genéticos de mutação, cruzamento e

seleção para evoluir a população de soluções potenciais do problema a ser resolvido em direção ao ótimo global, capaz de trabalhar de forma confiável em ambientes não lineares e multimodais (PRICE; STORN; LAMPINEN, 2005; NORMAN; IBA, 2008; ZHANG; SANDERSON, 2009a; LIU, YIN; GU, 2014).

O algoritmo de ED conquistou as primeiras colocações do *International Contest on Evolutionary Optimization* do IEEE (*Institute of Electrical and Electronics Engineers)* nos anos de 1996 e 1997 e continua a exibir desempenho notável em competições sobre diferentes tipos de problemas de otimização, tais como problemas dinâmicos, multi-objetivos, restritos e multimodais, realizadas em uma série de conferências sobre CE (QING; LEE, 2010; DAS; SUGANTHAN, 2011).

Por se mostrar uma abordagem competitiva com algoritmos de otimização mais complexos, apresentando precisão e velocidade convergência superiores a muitos outros métodos de otimização, a ED tornou-se uma técnica popular entre pesquisadores e vem sendo aplicada em diversas áreas da engenharia (MALLIPEDDI et al., 2011; TANABE; FUKUNAGA, 2014), tais como aerodinâmica (ROGALSKY; DERKSEN; KOCABIYIK, 1999), comunicação de satélites (SALMAN et al., 2010), engenharia química (VÁZQUEZ-OJEDA et al., 2013) e projetos de sistemas mecânicos (SARUHAN, 2014).

Embora originalmente desenvolvido para solução de problemas complexos de otimização irrestrita no domínio contínuo (ZHANG; SANDERSON, 2009a; RAMOS; TUPIA, 2013), ao longo dos anos, alguns pesquisadores vem demonstrando que a abordagem de ED é apropriada para otimização combinatória, além de ser eficaz e competitiva com outras abordagens neste domínio (TASGETIREN 2004; ONWUBOLU; DAVENDRA, 2006; ONWUBOLU; DAVENDRA, 2009; PRADO et al., 2010; RAMOS; TUPIA, 2013; LIU; YIN; GU, 2014; DONG, 2015; SHAO; PI, 2016).

Atualmente o algoritmo de ED tem sido bastante utilizado para resolver problemas de otimização em domínios discretos, tais como Problemas de Despacho, Problemas do Caixeiro Viajante, Problemas de Roteirização de Veículos e Problemas de Programação da Produção (PPP) (TIEN et al., 2015), foco deste estudo.

A Programação da Produção (PP), uma das atividades executadas pelo Planejamento, Programação e Controle da Produção (PPCP) constitui uma parte central dos processos associados à produção (CARVALHO et al., 2016) e representa um problema prático abordado, muitas vezes, pelas empresas

manufatureiras e prestadoras de serviço (PINEDO, 2008; MORAIS; MOCCELLIN, 2010; AKROUT et al., 2013a).

A PP tem um significativo efeito na manutenção da posição competitiva das empresas manufatureiras e prestadoras de serviço (LI; YIN, 2013; LIN et al., 2015) pois apresenta estreitas ligações com o desempenho das empresas no âmbito das dimensões estratégias como rapidez, confiabilidade, flexibilidade, qualidade e custos (SLACK; CHAMBERS; JOHNSTON, 2002).

A PP pode ser entendida como um conjunto de funções de tomada de decisão que envolve a alocação de recursos limitados a tarefas a serem processadas, bem como a sequencia de execução das tarefas, de modo a otimizar medidas ou critérios de desempenho pré-estabelecidos (BAKER, 1974; JOHNSON; MONTGOMERY, 1974; MACCARTHY; LIU, 1993; YANG; LIAO, 1999; PINEDO, 2008; KARIMI; DAVOUDPOUR, 2014; LIU; YIN; GU, 2014).

A PP é uma tarefa de alta complexidade e difícil resolução (GOYAL; SRISKANDARAJAH, 1988) que tem sido considerada um desafio permanente aos pesquisadores (GUIMARÃES et al., 2013) por envolver um grande número de possibilidades de solução, que acabam por despender um elevado tempo no processo de resolução (MORTON; PENTICO, 1993; AKROUT et al., 2013a; JARBOUI; SIARRY; TEGHEN, 2013).

Os Problemas de Programação da Produção (PPP) demonstram uma complexidade ainda maior quando se busca a solução ótima, pois de acordo com a teoria da complexidade computacional (GAREY; JOHNSON,1979) os PPPs são classificados como NP-*Hard* (*Non Polinominal Hard*), ou seja, são problemas para os quais não existem algoritmos capazes de obter a melhor solução em tempo polinomial (SIPSER, 2012).

Desde o trabalho pioneiro de Johnson em 1954, muitos esforços têm sido realizados para resolver os PPPs. No entanto, de acordo com Lin et al. (2015) não houve até o momento nenhum algoritmo que resolvesse completamente tais problemas, ou seja, nenhum algoritmo foi capaz de encontrar uma solução ótima global para todos os *Benchmarks* de problemas. Isto significa que é necessário determinar as melhores formas de resolver os PPPs.

Diante do exposto, busca-se nesta tese resolver o PPP em ambientes de produção *Flow Shop* Permutacional (FSP), por meio do desenvolvimento de novos métodos de solução baseados em abordagens de ED.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

Os Problemas de Programação da Produção (PPP) em ambientes *Flow Shop* Permutacional (FSP), que caracterizam um caso especial do *Flow Shop* (FS) (MOKHTARI; ABADI; CHERAGHALIKHANI, 2011; LIN et al., 2015), têm sido estudados por muitos pesquisadores devido a sua significância prática e acadêmica (LI; YIN, 2013; SHEN; WANG; WANG, 2015).

O FSP, considerado um dos mais importantes PPPs (FUCHIGAMI; RANGEL, 2015), está presente em indústrias de produção intermitente tais como indústrias químicas, eletrônicas, fundições, processamento de fibra de vidro, dentre outras (TASGETIREN et al., 2013).

Os PPPs em ambientes FSP, objeto deste estudo, consistem em determinar a ordem de *n* tarefas em *m* máquinas sequenciais para otimizar determinado(s) critério(s) de desempenho, quando todas as *n* tarefas são processadas nas *m* máquinas na mesma ordem (TAILLARD, 1993; PAN, TASGETIREN; LIANG, 2008; MOKHTARI; ABADI; CHERAGHALIKHANI, 2011; LI; YIN, 2013; LIN et al., 2015).

Os PPPs em ambientes FSP, devido a sua complexidade computacional, são problemas difíceis de resolver. Os PPPs somente podem ser resolvidos, eficientemente, de maneira ótima (solução exata) por meio de métodos de programação matemática, em casos de pequeno porte (NAGANO; BRANCO; MOCCELLIN, 2009; LIU et al., 2014).

No entanto, em casos de problemas de médio e grande porte, abordagens aproximadas para obtenção de boas soluções (mas não necessariamente as ótimas) em tempos computacionais razoáveis, são aceitáveis (MONTOYA, 2006; EMMONS; VAIRAKTARAKIS, 2013). Tal fato torna atrativo, para o campo das pesquisas, a proposição de novos métodos para solução do problema.

Diante do exposto, pode-se destacar a resolução dos PPPs em FSP usando técnicas da CE, particularmente eficazes em comparação com outros algoritmos de busca em grandes espaços de buscas típicos dos problemas de programação do mundo real (DAHAL; TAN; COWLING, 2007; MALLIPEDDI et al., 2011; TANABE; FUKUNAGA, 2014).

No contexto dos AEs, a ED cujos méritos compartilhados dos AEs são capacidade de busca global, versatilidade para problemas característicos e pouca demanda em uma boa solução inicial (QING; LEE, 2010), tem se mostrado capaz de

produzir resultados promissores quando aplicado a diferentes tipos de problemas complexos de otimização (FAN; ZHANG, 2016), tornando-se uma abordagem atraente para a solução dos PPPs, foco desta tese.

O algoritmo de ED tem como conceito básico a utilização de uma população de indivíduos, que representam soluções potenciais dentro do espaço de busca do problema a ser resolvido, e que evoluem ao longo de gerações por meio da utilização de operadores genéticos. A cada geração, esta população é eventualmente modificada dando lugar a uma nova população de mesmo tamanho constituída por novos indivíduos que se mostrem melhores ou mais promissores formando assim uma nova geração (STORN; PRICE, 1997; QIN; SUGANTHAN, 2005; BROWNLEE, 2011; SIMON, 2013; EIBEN; SMITH, 2015).

De acordo com Liu, Yin e Gu (2014) e Fu et al. (2017) a ideia crucial do algoritmo de ED é a estratégia mutante que gera novos indivíduos mutantes adicionando a diferença ponderada entre dois indivíduos a um terceiro indivíduo. Então, o operador de cruzamento específico é empregado para combinar o indivíduo mutante com o indivíduo principal para gerar descendentes individuais. Finalmente, os melhores indivíduos da população de progenitores e descendentes são selecionados para a próxima geração.

Diferentemente de outros AEs, o algoritmo de ED não usa funções de distribuição de probabilidade a fim de adicionar variações na população, ao invés disto, usa a diferença de vetores aleatoriamente selecionados como fonte de variações aleatórias (PRICE; STORN; LAMPINEN, 2005; ONWUBOLU; DAVENDRA, 2009), ou seja, o algoritmo de ED usa informações de distância e direção da população corrente para guiar o processo de busca (LIU; YIN; GU, 2014; FU et al., 2017).

O sucesso da ED se deve a uma autoadaptação implícita contida dentro da estrutura do algoritmo. Mais especificamente, como para cada solução candidata a regra de busca depende de outras soluções pertencentes da população, a capacidade de detectar novas soluções filhas promissoras depende da atual distribuição de soluções dentre o espaço de busca (FEOKTISTOV, 2006; NERI; TIRRONEN, 2010).

Storn e Price (1995) e Storn e Price (1997) desenvolveram dez estratégias mutantes que podem ser adotadas nos algoritmos de ED, no entanto, outras estratégias de ED têm sido propostas e adotadas no desenvolvimento dos métodos

de solução, dado que o algoritmo de ED pode apresentar diferentes comportamentos baseados na estratégia selecionada (CARVALHO et al., 2016). De acordo com Fu et al. (2017) o desenvolvimento de novas estratégias mutantes ou a introdução de novas técnicas para selecionar adaptativamente estratégias mutantes constitui uma das várias categorias de variantes de ED propostas para melhorar o desempenho do algoritmo de ED.

Outra importante característica da ED é a pequena quantidade de parâmetros utilizados, sendo eles a quantidade de vetores (indivíduos) mantidos na população (*Np*), o fator ou taxa de mutação (*F*) e o fator ou taxa de cruzamento (*CR*) (BROWLEE, 2011; DAS; SUGANTHAN, 2012; SIMON, 2013; EIBEN; SMITH, 2015; PEÑUÑURI et al., 2016). No entanto, o desempenho de um algoritmo de ED, assim como o de outros AEs, depende das configurações adequadas dos parâmetros (ZHANG; SANDERSON, 2009a; MALLIPEDDI et al., 2011; TANABE; FUKUNAGA, 2014; FAN; ZHANG, 2016) que tem forte influencia no desempenho de convergência do algoritmo (QIN; HUANG; SUGANTHAN, 2009; BANITALEBI; AZIS; AZIS, 2016; PIEREZAN et al., 2017).

No algoritmo de ED clássico os parâmetros de controle e as estratégias de mutação, são especificados pelo usuário e mantidos constantes durante a execução. Estes parâmetros são conhecidos por serem dependentes do problema, ou seja, a configuração dos parâmetros e a escolha dos melhores parâmetros de controle depende do problema de otimização (ZAMUDA; BREST, 2015; PIERZAN et al., 2017), sendo geralmente necessárias diversas tentativas para determinar os valores apropriados para cada problema específico, uma vez que nem sempre é uma tarefa fácil ajustar estes parâmetros de controle (ZHANG; SANDERSON, 2009a; QING; LEE, 2010; MALLIPEDDI et al., 2011; TANABE; FUKUNAGA, 2014; FU et al., 2017; PIERZAN et al., 2017).

De acordo com Banitalebi, Azis e Azis (2016) e Fan e Zhang (2016) foram realizados diversos estudos empíricos que introduzem várias orientações para selecionar os parâmetros de controle e as estratégias para melhorar o desempenho do algoritmo de ED (STORN; PRICE, 1997; GÄMPERLE; MÜLLER; KOUMOUTSAKOS, 2002; RONKKONEN; KUKKONEN; PRICE, 2005; MEZURA-MONTES; VELAZQUEZ-REYES; COELLO, 2006; ZAHARIE, 2009). No entanto, quando as propriedades dos problemas de otimização são desconhecidas, é difícil escolher parâmetros de controle e estratégias apropriadas para o algoritmo de ED,

dado que as diretrizes propostas por tais estudos geralmente são justificativas insuficientes porque se baseiam em experimentos específicos (FAN; ZHANG, 2016).

Para evitar a necessidade de ajustes de parâmetros e seleção de estratégias dependentes do problema, na literatura especializada têm sido introduzidos diferentes mecanismos adaptativos para atualizar dinamicamente os parâmetros de controle sem um conhecimento prévio do relacionamento entre as configurações e as características do problema (ZHANG; SANDERSON, 2009a; MALLIPEDDI et al., 2011; TANABE; FUKUNAGA, 2014; BANITALEBI; AZIS; AZIS, 2016; FAN; ZHANG, 2016).

Zhang e Sanderson (2009a) e Fu et al. (2017) ressaltam que é possível conseguir melhorias na performance do algoritmo de ED, em termos de robustez e taxa de convergência, criando mecanismos adaptativos ou autoadaptativos para ajustar seus parâmetros de controle. O desenvolvimento de variantes adaptativas e autoadaptativas não só reduzem a necessidade de parâmetros ótimos, mas também estendem a eficiência computacional do algoritmo de ED.

A ideia básica nas abordagens adaptativas, inicialmente introduzidas para Programas Evolutivos (Programação Evolucionária), consiste em alterar os parâmetros de controle durante a evolução com base em realimentações (*feedbacks*) da busca, enquanto que nas abordagens autoadaptativas, os parâmetros do algoritmo são anexados a indivíduos e otimizados em conformidade, ou seja, os parâmetros evoluirão durante a evolução através de uma inclusão na codificação genética que ela própria é submetida ao processo evolutivo (EIBEN; HINTERDING; MICHALEWICZ, 1999; TEO, 2006; BANITALEBI; AZIS; AZIS, 2016).

As experiências em aplicações do algoritmo de ED (ONWUBOLU; DAVENDRA, 2009; SUN et al., 2011; TONGE; KULKARNI, 2012; GUIMARÃES et al., 2013; JARBOUI; SIARRY; TEGHEN, 2013; ALLAHVERDI, 2015; e ALLAHVERDI, 2016) revelaram ser promissor seu uso para solução de PPPs, foco desta tese. No entanto, o desempenho da busca do algoritmo de ED ainda precisa ser melhorado, uma vez que os modernos problemas de otimização estão se tornando cada vez mais complicados (FAN; ZHANG, 2016).

Em relação ao desenvolvimento de algoritmos de ED para solução de PPPs, pode ser verificado na literatura especializada que as adaptações do algoritmo de ED, em grande parte, se limitam as fases de inicialização da população e de exploração do espaço de busca (PRADO et al., 2010; CARVALHO et al., 2016), o

que reforça a importância de mais investigações e motiva o desenvolvimento de novos métodos para resolução dos PPP, utilizando abordagens de ED.

Destaca-se que na revisão de literatura realizada para o desenvolvimento desta tese, que cobriu 40 trabalhos orientados a solução dos PPPs em FSP utilizando abordagens de ED, somente quatro trabalhos apresentaram abordagens de ED autoadaptativas, isto é, algoritmos em que o ajuste dos parâmetros do algoritmo ocorre dinamicamente durante o processo evolutivo. No entanto, somente no trabalho de Xu, Xiang e Wang (2010) os parâmetros *F* e *CR* são ambos ajustados de forma autoadapta. O estudo de Li e Yin (2013) utiliza mecanismo autoadaptativo para ajuste do parâmetro *CR*, enquanto que nos estudos de Santuci, Baioletti e Milani (2014) e Santuci, Baioletti e Milani (2016) mecanismos autoadaptativos são utilizados para ajuste do parâmetro *F*.

Assim, diante do exposto, este trabalho visa contribuir para um campo das pesquisas ainda pouco explorado por meio do desenvolvimento de novos algoritmos de ED Autoadaptativa para o problema de minimização do *Makespan* em PPPs em ambientes FSP. Os mecanismos autoadaptativos de ajuste dos parâmetros *CR* e *F* e de seleção de estratégias de mutação são incorporados nos algoritmos propostos com o intuito de melhorar a qualidade da solução, acelerar a convergência do algoritmo e ainda preservar as principais características do algoritmo de ED.

1.2 OBJETIVOS DA TESE

Esta tese tem como objetivo geral o desenvolvimento de novos métodos de solução para a minimização do *Makespan* em Problemas de Programação da Produção em ambientes *Flow Shop* Permutacional, baseado(s) em abordagens de Evolução Diferencial Autoadaptativa.

Os objetivos específicos desta tese são:

i. Estudar o problema de programação da produção em ambientes *Flow Shop* Permutacional;

ii. Estudar o algoritmo de Evolução Diferencial;

iii. Explorar os enfoques de Evolução Diferencial Adaptativa/Autoadaptativa e de Evolução Diferencial Discreta;

iv. Realizar um levantamento, na literatura especializada, de trabalhos que contemplem o desenvolvimento de métodos baseados em abordagens de Evolução Diferencial para a programação da produção em *Flow Shop* Permutacional;
v. Desenvolver e implementar em linguagem computacional novos métodos baseados em Evolução Diferencial Autoadaptativa para o caso de Minimização do *Makespan* em ambientes *Flow Shop* Permutacional;
vi. Testar os algoritmos em problemas específicos para ambientes *Flow Shop* Permutacional disponíveis na literatura especializada em problemas teste (*Benchmarks*);
vii. Avaliar a qualidade das soluções fornecidas pelos algoritmos propostos em termos de erro relativo percentual em relação aos limitantes superiores (*Makespan*) da função objetivo disponíveis na literatura especializada;
viii. Avaliar o desempenho dos algoritmos de ED propostos em termos de desvio padrão, convergência e tempo computacional para 50 simulações.

1.3 CONTRIBUIÇÕES DA TESE

Na literatura especializada, conforme já mencionado poucos trabalhos tratam do PPP em FSP por meio de abordagens de Evolução Diferencial Autoadaptativa. Portanto, a principal contribuição pretendida nesta tese para a área de conhecimento consiste no desenvolvimento de novos métodos de solução para o PPP em FSP baseados em abordagens autoadaptativas, ou seja, o desenvolvimento de métodos em que o ajuste dos parâmetros de controle do algoritmo de ED e a seleção das estratégias de mutação ocorrem dinamicamente durante o processo evolutivo.

Nesta tese, além de uma versão discreta do Algoritmo de Evolução Diferencial desenvolvida para a minimização do *Makespan* em problemas de programação da produção em ambientes *Flow Shop* Permutacional, duas novas variantes do algoritmo de Evolução Diferencial Autoadaptativa são desenvolvidas. Ressalta-se que as variantes autoadaptativas desenvolvidas nesta tese, são inéditas para o caso de programação da produção em ambientes *Flow Shop* Permutacional.

As duas novas variantes do algoritmo de Evolução Diferencial Autoadaptativa desenvolvidas nesta tese foram baseadas no algoritmo JADE (*J Adaptive Differential Evolution*) proposto por Zhang e Sanderson (2007) e Zhang e Sanderson (2009b) e no algoritmo SaDE (*Self-adaptive Differential Evolution*) proposto por Qin e Suganthan (2005) e Qin, Huang e Suganthan (2009).

Na primeira variante desenvolvida são incorporados mecanismos autoadaptativos de ajuste dos parâmetros *CR* e *F*, além de uma nova estratégia de mutação. Na segunda variante desenvolvida são incorporados mecanismos autoadaptativos de ajuste dos parâmetros *CR* e *F* e de seleção de estratégias de mutação.

1.4 ORGANIZAÇÃO DA TESE

O restante desta tese está organizado em sete capítulos sendo que este, o primeiro foi direcionado a estabelecer uma introdução geral.

O Capítulo 2 contempla os fundamentos dos problemas de programação da produção, bem como o caso específico do problema de programação da produção em ambientes *Flow Shop*, de interesse particular deste estudo, o problema de programação da produção em *Flow Shop* Permutacional. Enquanto, o Capítulo 3 é dedicado a elucidar os conceitos inerentes ao algoritmo de Evolução Diferencial, bem como os conceitos de Evolução Diferencial Adaptativa/Autoadaptativa e de Evolução Diferencial Discreta.

No Capítulo 4 uma revisão de literatura acerca de trabalhos que abordam a temática desta tese, ou seja, métodos baseados em Evolução Diferencial para solução do problema de programação em *Flow Shop* Permutacional é apresentada.

No Capítulo 5 são apresentadas para o caso de minimização do *Makespan* em problemas de programação da produção em ambientes *Flow Shop* Permutacional, uma versão discreta do algoritmo de Evolução Diferencial e duas variantes do algoritmo de Evolução Diferencial Autoadaptativa.

No Capítulo 6 são explicitados os casos selecionados para testar os algoritmos propostos, bem os procedimentos para geração e análise dos dados. E no Capítulo 7 os resultados são apresentados e discutidos.

O Capítulo 8 contém as conclusões da pesquisa, alguns temas que podem nortear futuras pesquisas, além das publicações relacionadas à tese.

Capítulo 2 – FUNDAMENTOS DE PROGRAMAÇÃO DA PRODUÇÃO

Este capítulo visa retratar o problema de programação da produção abordado neste estudo. Serão expostos alguns conceitos fundamentais sobre os problemas de programação da produção, bem como, um caso específico do problema de programação da produção em ambientes *Flow Shop*, de interesse particular deste estudo, o problema de programação da produção em *Flow Shop* Permutacional.

2.1 PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO

A Programação da Produção (PP) pode ser definida como a alocação de recursos através do tempo para a realização de tarefas, para melhor satisfazer critério(s) de desempenho(s) pré-definido(s) (BAKER, 1974; MACCARTHY; LIU, 1993; YANG; LIAO, 1999; PINEDO, 2008), ou seja, a PP consiste na determinação da sequência em que as máquinas irão processar as tarefas, em um determinado período de tempo, de modo a otimizar alguma medida de desempenho (JOHNSON; MONTGOMERY, 1974).

Um Problema de Programação da Produção (PPP) pode ser entendido como um problema de n tarefas $\{J_1, J_2, ... , J_j, ... , J_n\}$ que devem ser processadas em m máquinas $\{M_1, M_2, ... , M_k, ... , M_m\}$ que estão disponíveis (GRAHAM et al., 1979; MACCARTHY; LIU, 1993; TAILLARD, 1993).

Em PPPs o processamento de uma tarefa i ($1 \leq i \leq n$) em uma máquina j (($1 \leq j \leq m$) pode ser definido como uma operação, designada de op_{ij}, que por sua vez possui um tempo t_{ij} associado (MACCARTHY; LIU, 1993). Para cada tarefa i deve existir uma data de liberação (*release date* ou *release time* ou *ready time* – l_i) que corresponde à data a partir da qual a tarefa pode ser executada e uma data de entrega (*due date* - d_i), que corresponde à data em que a tarefa deve estar concluída (TAILLARD, 1993). O fluxo padrão das tarefas ou a ordem das máquinas podem ou não ser fixado para algumas ou todas as tarefas (MACCARTHY; LIU, 1993).

De acordo com MacCarthy e Liu (1993) um PPP pode ser classificado de acordo com os ambientes de máquinas, as características de processamento e restrições e o critério de desempenho ou critério de otimalidade considerado.

Os PPPs podem ser representados por uma notação, expressa por três campos *α*/*β*/*γ* (GRAHAM et al., 1979; LAWLER et al., 1989; PARKER, 1995; T´KINDT; BILLAUT, 2002; PINEDO, 2008), em que: *α* especifica os ambientes de máquinas; *β* especifica as características de processamento e restrições; e *γ* especifica os critérios de desempenho ou otimalidade.

Os diferentes campos desta notação representam as variáveis que definem o problema de programação sob investigação, as quais são discutidas em detalhes nas subseções seguintes.

2.1.1 Ambientes de Máquinas

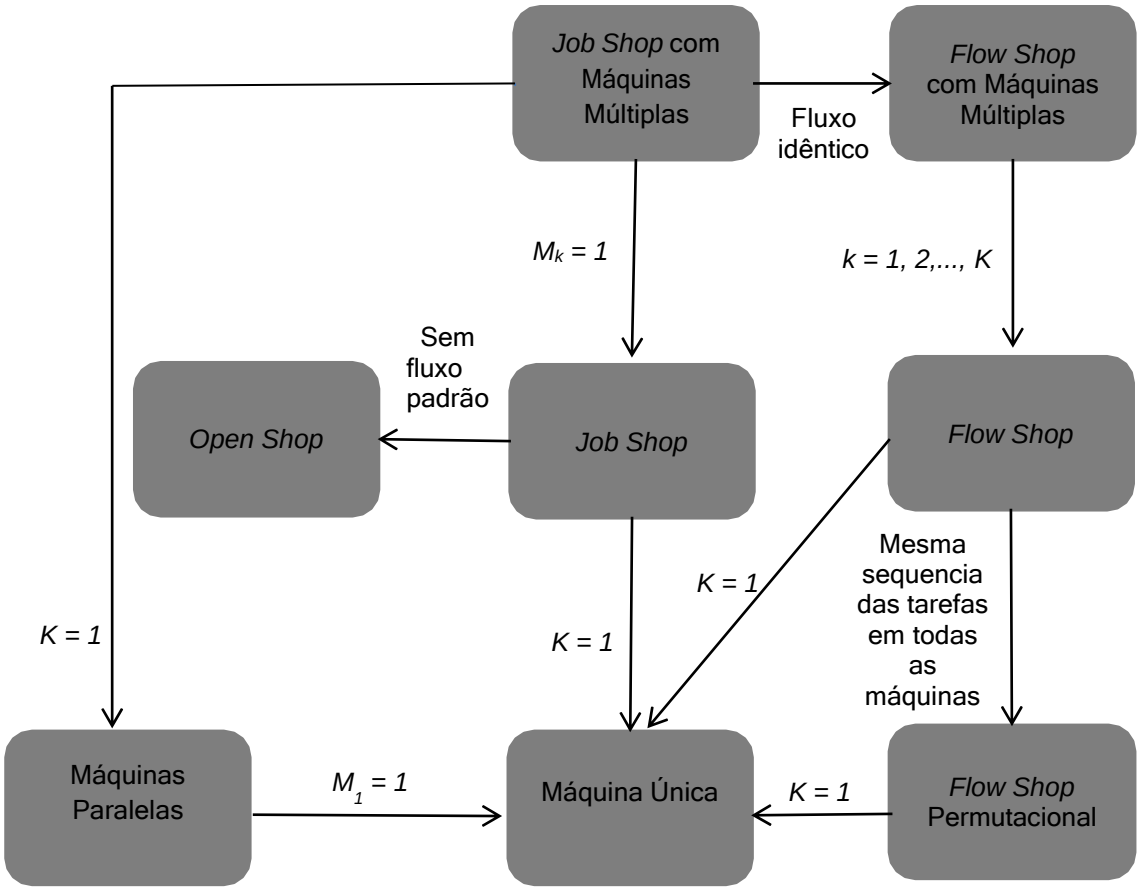
Os possíveis ambientes de máquinas verificados nos PPPs, com base em Graham et al. (1979), MacCarthy e Liu (1993) e Pinedo (2008) são classificados em:

i. Máquina Única (1 ou ϕ): Há apenas uma máquina disponível para processamento das tarefas;
ii. Máquinas Paralelas (*P*): Há *m* máquinas em paralelo, sendo que uma tarefa *j* pode ser processada em qualquer uma das *m* máquinas. As máquinas em paralelo podem ser de três diferentes tipos, conforme segue:
    ii.1) Máquinas Idênticas em Paralelo (*Pm*): Os tempos de processamento das tarefas são os mesmos para todas as máquinas;
    ii.2) Máquinas Proporcionais ou Uniformes em Paralelo (*Qm*): Os tempos de processamento das tarefas variam de acordo com um fator de proporcionalidade entre as máquinas; e
    ii.3) Máquinas Não-Relacionadas em Paralelo (*Rm*): Os tempos de processamento das tarefas variam entre as máquinas de forma arbitrária.
iii. *Flow Shop* (*Fm*): Há *m* máquinas em série e todas as tarefas devem ser processadas em cada uma das *m* máquinas, seguindo o mesmo fluxo de processamento nas máquinas;
iv. *Flow Shop* Permutacional (*PFm):* É um *Flow Shop*, em que a ordem de processamento de todas as tarefas nas *m* máquinas é a mesma em todas as máquinas *do Flow Shop*;

v. *Flow Shop* Flexível (*FFc*): É uma generalização dos ambientes *Flow Shop* e Máquinas em Paralelo, em que, em pelo menos um dos estágios de produção existe um conjunto de máquinas em paralelo;
vi. *Job Shop* (*Jm*): Há *m* máquinas, no entanto, cada tarefa tem sua própria ordem de processamento nas máquinas, ou seja, seu próprio roteiro de produção;
vii. *Job Shop* Flexível (*FFc*): É uma generalização dos ambientes *Job Shop* e Máquinas em Paralelo, em que, em pelo menos um dos estágios de produção existe um conjunto de máquinas em paralelo; e
viii. *Open Shop* (*Om*): Há *m* máquinas, no entanto, não há fluxo definido (específico) para as tarefas serem processadas nas máquinas.

Os diversos ambientes de máquinas e seus relacionamentos são ilustrados na Figura 2.1.

Figura 2.1 – Ambientes de Máquinas e seus Relacionamentos em Problemas de Programação da Produção



K - Número de estágios de produção
M_k - Número de máquinas do estágio *k* (com *k = 1, 2, ..K)*

Fonte: Adaptado de MacCarthy e Liu (1993)

Conforme os ambientes de máquinas anteriormente explicitados, o parâmetro $\alpha \in$ {1 ou ϕ, *P (Pm*, *Qm*, *Rm)*, *Fm*, *FFc*, *Jm*, *FJc*, *Om*}.

A fim de acomodar outros parâmetros do problema, como o número de estágios, tipo de máquinas em cada estágio e número de máquinas por estágio. O parâmetro α foi expandido por Vigner, Billaut e Proust (1999) passando a ser representado por (α_1, α_2, α_3, α_4), em que: α_1 representa os ambientes de máquinas (1 ou ϕ, *P (Pm*, *Qm*, *Rm)*, *Fm*, *FFc*, *Jm*, *FJc*, *Om*); α_2 representa o número de estágios (M_k), quando este for maior que 1; α_3 representa o tipo de máquinas em paralelo (*Pm*, *Qm*, *Rm*); e α_4 representa o número de máquinas em cada estágio.

2.1.2. Características de Processamento e Restrições

Na literatura pode ser verificado um grande número de características relativas ao processamento e restrições que podem ser verificadas nos diferentes PPPs. De acordo com Allahverdi, Cheng e Kovalyov (2008) e Pinedo (2008) algumas características que constituem entradas para o campo β são:

i. Datas de Liberação (r_j): Datas a partir da qual a tarefa pode ser executada. Se uma data de liberação para uma tarefa não for definida, a mesma pode ter seu processamento iniciado em qualquer tempo;
ii. Precedência (*prec*): Indica que uma tarefa deve ser concluída antes que outra tarefa tenha seu processamento iniciado;
iii. Interrupções (*prmp*): Situação em que é permitido interromper o processamento de uma tarefa a qualquer momento e processar uma tarefa diferente na máquina. O processamento já realizado da tarefa na máquina não é perdido;
iv. Bloqueio (*block*): Caso em que uma tarefa após a conclusão de seu processamento numa máquina bloqueia a máquina se a máquina seguinte estiver ocupada processando outra tarefa;
v. *No-Wait* (*nwt*): Situação em que não é permitido que tarefas esperem entre duas máquinas sucessivas. Isso implica que o tempo de inicio de uma tarefa no primeiro estágio tem que ser atrasado para garantir que a tarefa possa passar pelo sistema sem ter que esperar em qualquer máquina;
vi. Elegibilidade de Máquinas (M_j): Este caso aparece em ambientes de máquinas paralelas e *flow shop* flexível e ocorre quando uma tarefa deve ser processada em uma máquina ou tipo de máquina específico;
vii. Permutação (*prmu*): Um caso que pode aparecer em ambientes *Flow Shop* e consiste no fato de que cada tarefa deve ser processada em todas as *m* máquinas seguindo a mesma ordem;
viii. *Setup* Dependente da Sequência (s_{ij}): Em problemas de programação pode haver casos em os *setups* para processamento das tarefas são dependentes da sequencia;
ix. Processamento em Lotes (*batch*): Caso uma máquina seja capaz de processar um número de tarefas simultaneamente, as tarefas são agrupadas para processamento em lotes; e

x. Recirculação (*rcrc*): Caso que pode ocorrer em ambientes *Job Shop* e *Job Shop* Flexível, quando uma tarefa pode visitar uma máquina mais de uma vez.

Conforme as características de processamento e restrições dos problemas de programação da produção anteriormente mencionadas o parâmetro $\beta \in \{r_j,\ prec,\ prmp,\ block,\ nwt,\ M_j,\ prmu,\ s_{ij},\ batch,\ rcrc\}$.

Em função da diversidade de características não elencadas nesta seção, mas presentes em muitos problemas de programação, este conjunto pode ser expandido.

2.1.3. Critérios de Desempenho ou Otimalidade

A PP é realizada buscando atingir um critério, ou um conjunto de critérios de desempenho. Os critérios de desempenho, que segundo Mesquita et al. (2008) possuem a função de avaliar a qualidade de uma determinada solução, estão relacionados com os objetivos de desempenho da produção – qualidade, flexibilidade, rapidez, confiabilidade e custo (SLACK; CHAMBERS; JOHNSTON, 2002).

Segundo French (1982), Bedworth e Bailey (1987), MacCarthy e Liu (1993), Morton e Pentido (1993) e Pinedo (2008), os critérios de otimização mais comumente adotados em PPPs são:

i. Data de término da *j-ésima* tarefa (*Completion Time* - C_j): corresponde ao período desde o início da programação na data zero até ao momento em que a tarefa *j* é finalizada.
ii. Duração total da programação (*Makespan* - C_{max}): corresponde ao máximo ($C_1, C_2, ..., C_j$), que é equivalente ao tempo de conclusão da última tarefa a deixar o sistema.
iii. Tempo de fluxo da *j-ésima* tarefa (*Flow Time* - F_j): é o tempo entre o momento que a tarefa está disponível para ser processada e o momento em que ela é completada, correspondendo, então, ao tempo que a tarefa *j* permanece no sistema. O tempo de fluxo pode ser matematicamente expressado como:

$$F_j = C_j - I_j \qquad (2.1)$$

iv. Atraso da *j-ésima* tarefa (*Lateness* - L_j): corresponde ao desvio entre o tempo de conclusão da tarefa e sua data de entrega. O *lateness* de uma tarefa *j* é definido por:

$$L_j = C_j - d_j \tag{2.2}$$

v. Adiantamento da *j-ésima* tarefa (*Earliness* – E_j): Uma tarefa pode ter um *lateness* positivo se for completada depois da *due date* e um *lateness* negativo ou *earliness* (E_j) se for completada antes. O *earliness* de uma tarefa pode ser dado por:

$$E_j = \max(d_j - C_j, 0) \tag{2.3}$$

vi. Atraso da *j-ésima* tarefa (*Tardiness* - T_j): corresponde ao atraso, na execução da tarefa, em relação à sua data de entrega. O *Tardiness* de uma tarefa *j* é expressado matematicamente como:

$$T_j = \max(C_j - d_j, 0) = \max(L_j, 0) \tag{2.4}$$

Pinedo (2008) destaca que a diferença entre o *Tardiness* e o *Lateness* reside no fato de que o *Tardiness* nunca é negativo. A unidade de penalização de uma tarefa é definida conforme segue:

$$U_j = \begin{cases} 1, \text{se} C_j > d_j \\ 0, \text{casocontrário} \end{cases} \tag{2.5}$$

Outros critérios de otimização adotados em Programação da Produção, de acordo com Bedworth e Bailey (1987), MacCarthy e Liu (1993 e Allahverdi, Cheng e Kovalyov (2008) são listados no Quadro 2.1.

Quadro 2.1 – Critérios de Otimização Adotados em Problemas de Programação da Produção

Notação	Descrição
$\Sigma C_i/n$	Duração Média da Programação
$\sum C_j$	Tempo Total de Conclusão
$\sum w_j C_j$	Tempo Total de Conclusão Ponderado
ΣF_j	Tempo Total de Fluxo
$\Sigma w_j F_j$	Tempo de Fluxo Total Ponderado
$\Sigma F_j/n$	Tempo Médio de Fluxo
$\Sigma W_j/n$	Tempo Médio de Espera
$\Sigma w_j W_j$	Tempo Total de Espera Ponderado
ΣW_j	Tempo Total de Espera
L_{max}	Adiantamento Máximo
ΣL_j	Adiantamento Total
$\Sigma w_j L_j$	Adiantamento Total Ponderado
T_{max}	Atraso Máximo
ΣT_j	Atraso Total
$\Sigma T_j/n$	Atraso Médio das Tarefas
$\Sigma w_j T_j$	Atraso Total Ponderado
ΣU_j	Número de Tarefas em Atraso

Fonte: o autor, 2017.

De acordo com os critérios anteriormente explicitados, o parâmetro $\gamma \in \{C_j, C_{max}, \Sigma C_j/n, \Sigma C_j, \Sigma w_j C_j, F_j, \Sigma F_j, \Sigma F_j/n, \Sigma w_j F_j; L_j, L_{max}, \Sigma L_j, \Sigma w_j L_j, T_j, T_{max}, \Sigma T_j; \Sigma T_j/n, \Sigma w_j T_j, E_j, E_{max}, \Sigma E_j, \Sigma E_j/n, \Sigma w_j E_j, W_j, \Sigma W_j, \Sigma W_j/n, \Sigma w_j W_j, \Sigma W_j/n, \Sigma U_j, \Sigma U_j/n\}$.

Os diferentes critérios de desempenho, anteriormente apresentados, relacionam-se a três tipos de tomada de decisão que, segundo Baker (1974), normalmente prevalecem na programação: utilização eficiente dos recursos; resposta rápida à demanda; e adaptação a prazos de entregas prescritos, que correspondem ao prazo de entrega final de uma tarefa que, se for atingido, anula o processamento já realizado.

Além dos critérios de desempenho anteriormente mencionados, diferentes critérios podem ser encontrados na literatura especializada, dentre os quais são destacados: Custo Total de *Setup* (*TSC*); Tempo Total de *Setup* (*TST* ou $\sum S_j$); Custos de Estoque (*CE*); Custos de Transporte (*CT*); Custo Total de Oportunidade

(*TCO*); Custo Total de Utilidade (*TCU*); Tempo Ocioso Total (*MIT*), Redução do Estoque em Processo (*WIP*) e Tempo de Bloqueio das Máquinas (*MBT*).

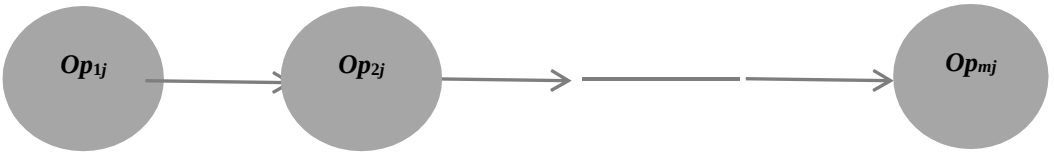
Como nos casos dos parâmetros α e β, em função da diversidade de critérios de desempenho utilizados em problemas de programação, verificados na literatura especializada, o conjunto de parâmetros γ pode ser expandido.

2.2 PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO EM AMBIENTES *FLOW SHOP*

Um *Flow Shop* é caracterizado por um fluxo de tarefas ininterrupto e contínuo através de múltiplas máquinas em série (GUPTA; STAFFORD JR., 2006). No *Flow Shop* o fluxo de trabalho é unidirecional, dado que todas as tarefas têm um fluxo padrão idêntico nas máquinas, ou seja, as tarefas possuem o mesmo roteiro de produção em todos os estágios de produção (TAILLARD, 1993).

Cada tarefa no *Flow Shop* requer uma sequencia específica de operações a serem cumpridas para a tarefa seja completada. De acordo com Baker (1974) no *Flow Shop* cada operação, depois da primeira, tem uma operação precedente direta, e cada operação, antes da última, tem uma operação sucessora direta. Este tipo de estrutura pode ser denominado estrutura de precedência linear, conforme mostrado na Figura 2.2.

Figura 2.2 – Estrutura de Precedência Linear para Determinada Tarefa J_j



Fonte: Boiko (2008)

Diante do exposto, pode-se afirmar, conforme Taillard (1993), que as operações de todas as *n* tarefas em um *Flow Shop* devem ser processadas nas *m* máquinas $\{M_1, M_2, ..., M_k, , M_m\}$, nesta ordem.

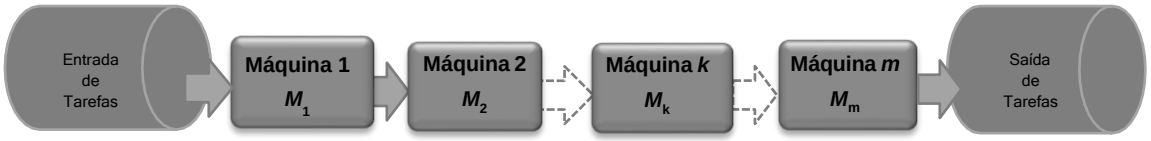
O PPP tratado neste trabalho é o *Flow Shop* Permutacional, um caso especial do *Flow Shop*, em que a ordem de processamento de todas as *n* tarefas nas *m*

máquinas é a mesma em todas as máquinas do *Flow Shop* (TAILLARD, 1993), ou seja, a permutação ou sequencia de tarefas fixada na primeira máquina é mantida para todas as máquinas (TASGETIREN et al., 2011).

O *Flow Shop* Permutacional é um típico problema de otimização combinatória, conhecidamente de natureza *NP-Hard* (*Non Polinominal Hard*), cuja solução consiste em determinar dentre as *n*! sequencias possíveis de tarefas, a sequencia que otimize uma determinada medida de desempenho (NAGANO; BRANCO; MOCCELLIN, 2009; MAKHTARI; ABADI; CHERAGHALINKHANI, 2011; LIN et al., 2015).

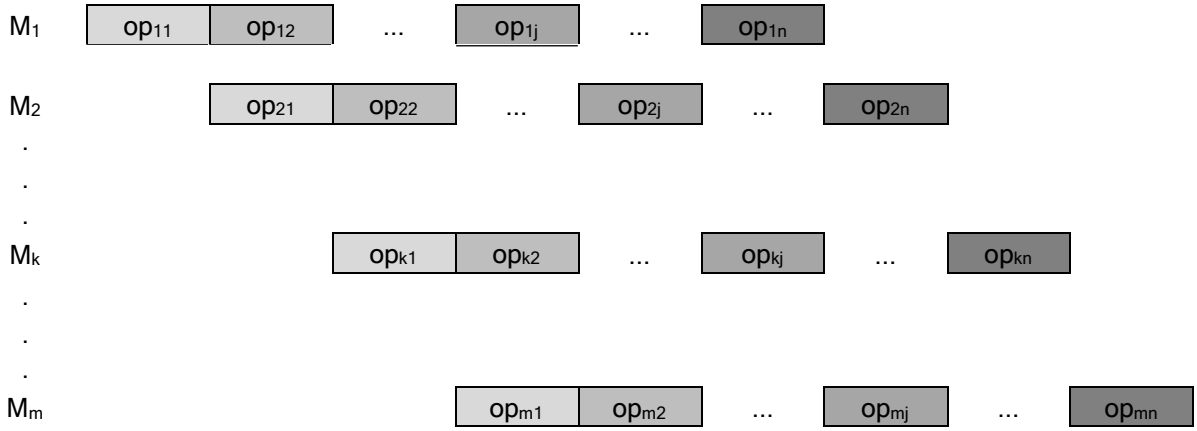
A Figura 2.3 ilustra um ambiente de produção *Flow Shop* Permutacional e a Figura 2.4 ilustra graficamente a programação de *n* tarefas e *m* máquinas em um ambiente *Flow Shop* Permutacional.

Figura 2.3 – Ambiente de Produção *Flow Shop* Permutacional



Fonte: o autor, 2017.

Figura 2.4 – Gráfico de Gantt da Programação de *n* Tarefas e *m* Máquinas em um Ambiente *Flow Shop* Permutacional



Fonte: o autor, 2017.

De acordo com Ahmadizar, Barzinpour e Arkat (2007), Sayadi, Ramezanian e Ghaffari-Nasad (2010), Liu e Liu (2013), Karimi e Davoudpour (2014) e Lin et al. (2015) no *Flow Shop* Permutacional, ambiente considerado neste estudo, as hipóteses adotadas são:

i. Cada tarefa tem exatamente uma operação a ser processada em cada máquina;
ii. Cada tarefa tem a mesma sequência na máquina e a sequência na qual cada máquina processa todas as tarefas é idêntica em todas as máquinas;
iii. Interrupções não são permitidas;
iv. Cada máquina pode processar no máximo, uma tarefa de cada vez e cada tarefa pode ser processada em no máximo uma máquina de cada vez;
v. Todas as tarefas estão disponíveis e prontas para inicio de processamento na data zero;
vi. Os tempos de *setup* são independentes da sequencia e incluídos nos tempos de processamento;
vii. Cada tarefa tem um tempo de processamento associado a cada máquina; e
viii. Uma sequencia é uma permutação de tarefas que representa a ordem na qual as tarefas serão processadas em todas as máquinas.

Baseado nas hipóteses anteriormente listadas, uma formulação matemática para minimização do *Makespan* em ambientes *Flow Shop* Permutacional, proposta por Sayadi, Ramezanian e Ghaffari-Nasad (2010) é apresentada.

Os índices, os parâmetros e as variáveis de decisão usados na formulação são apresentados no Quadro 2.2.

Quadro 2.2 - Índices, Parâmetros e Variáveis de Decisão da Formulação Matemática do Problema de Programação da Produção de *Flow Shop* Permutacional

Índices	$i = 1,\ldots m$ - Índices das máquinas $j=1,\ldots,n$ - Índices das tarefas $k=1,\ldots,n$ - Índices das posições
Parâmetros	n - Número de tarefas m - Número de estágios ou máquinas t_{ij} - Tempo de processamento da *j-ésima* tarefa na *i-ésima* máquina
Variáveis de Decisão	q_{ijk} - Instante de conclusão da *j-ésima* tarefa na *k-ésima* na posição na *i-ésima* máquina X_{jk} - Variável binária, que assume 1 se a *j-ésima* tarefa é processada na *k-ésima* posição ou 0 em caso contrário

Fonte: Sayadi, Ramezanian e Ghaffari-Nasad (2010)

As expressões que definem o modelo matemático proposto por Sayadi, Ramezanian e Ghaffari-Nasad (2010) para minimização do *Makespan* em ambientes FSP são listadas a seguir.

$$\text{Minimizar} \sum_{j=1}^{n} q_{mjn} X_{jn} \tag{2.6}$$

$$\sum_{k=1}^{n} (q_{(i+1)jk} - t_{(i+1)j}) X_{jk} \geq \sum_{k=1}^{n} q_{ijk} X_{jk}, i = 1,\ldots, m-1; j = 1,\ldots, n \tag{2.7}$$

$$\sum_{j=1}^{n} (q_{ij(k+1)} - t_{(ij)}) X_{j(k+1)} - \sum_{j=1}^{n} q_{ijk} X_{jk} \geq 0,\ i = 1,\ldots, m; k = 1,\ldots, n-1 \tag{2.8}$$

$$\sum_{j=1}^{n} X_{jk} = 1, \forall k \tag{2.9}$$

$$\sum_{k=1}^{n} X_{jk} = 1, \forall j \tag{2.10}$$

$$q_{ijk} \leq M X_{jk}, \forall i, j, k \tag{2.11}$$

$$q_{ijk} \geq 0, \forall i, j, k \tag{2.12}$$

$$X_{ijk} = \{0,1\}, \forall i, j, k \tag{2.13}$$

A função objetivo (equação 2.6) representa a minimização do *Makespan*. As restrições (equação 2.7 e equação 2.8) garantem que cada tarefa não terá seu processamento iniciado em uma máquina até ter tido concluído seu processamento na máquina anterior e seu predecessor ter tido seu processamento concluído naquela máquina. As restrições (2.9) garantem que cada posição da sequência é preenchida com apenas uma tarefa e as restrições (2.10) garantem que cada tarefa

é atribuída a apenas uma posição na sequência. O conjunto de restrições (2.11) é uma relação entre as variáveis binárias e as variáveis dos tempos de conclusão das tarefas. Quando cada variável binária torna-se zero, então a variável de tempo de conclusão será também igual a zero. Nas restrições (2.11) *M* assume um valor grande para evitar que quando uma variável binária assuma valor zero a variável de tempo de conclusão correspondente assuma valor diferente de zero. As restrições (2.12) e (2.13) indicam o domínio das variáveis (SAYADI; RAMEZANIAN; GHAFFARI-NASAD, 2010).

Para o problema de programação a produção em ambientes *Flow Shop* Permutacional um valor suficientemente grande para *M* pode ser obtido por meio do somatório dos tempos de processamento (t_{ij}) de todas as tarefas as *n* tarefas nas *m* máquinas.

Capítulo 3 - EVOLUÇÃO DIFERENCIAL

Este capítulo tem o propósito de apresentar conceitos sobre os Algoritmos de Evolução Diferencial, necessários ao desenvolvimento desta tese. Este capítulo contempla os fundamentos dos Algoritmos de Evolução Diferencial, bem como o Algoritmo de Evolução Diferencial Clássico. Também são apresentados neste capítulo os conceitos de Evolução Diferencial Adaptativa/Autoadaptativa e de Evolução Diferencial Discreta.

3.1 FUNDAMENTOS

O algoritmo de ED, introduzido por Storn e Price (1995) e Storn e Price (1997) para otimização de funções contínuas não lineares e não diferenciáveis, é como quase todos os AEs, um otimizador populacional que inicia o processo de otimização por amostragem do espaço de busca em múltiplos pontos iniciais escolhidos aleatoriamente (isto é, uma população de vetores representando os indivíduos ou soluções candidatas) (PRICE; STORN; LAMPINEN, 2005; ONWUBOLU; DAVENDRA, 2009; ZHANG; SANDERSON, 2009a).

O ED é, por natureza, um otimizador de função contínua sem derivada, pois codifica os parâmetros como números de ponto flutuante e os manipula com operações aritméticas simples, como adição, subtração e multiplicação. Como outros AEs, o ED gera novos pontos que são as perturbações/mutações dos pontos existentes (ZHANG; SANDERSON, 2009a).

O algoritmo de ED envolve a manutenção de uma população de soluções candidatas sujeita a iterações de cruzamento, avaliação e seleção. A abordagem de cruzamento envolve a criação de novos componentes de solução candidatos com base na diferença ponderada entre dois membros (indivíduos) da população selecionados aleatoriamente com distribuição uniforme e adicionados a um terceiro membro da população. Isso perturba os membros da população em relação à disseminação da população em geral. Em conjunto com a seleção, o efeito de perturbação auto-organiza a amostragem do espaço do problema, limitando-o a áreas de interesse conhecidas (BRONWLEE, 2011).

De acordo com Pan, Tasgetiren e Liang (2007) o algoritmo de ED primeiro muta uma população alvo para produzir uma população mutante, que então é recombinada com a população alvo de modo a gerar uma população teste. Finalmente, um operador de seleção é aplicado nas populações alvo e teste para determinar, baseado nas avaliações de aptidão, qual irá sobreviver para a próxima geração.

Em outras palavras, no algoritmo de ED um vetor (pai) na população é transformado por meio de uma diferença escalonada de outros vetores individuais selecionados aleatoriamente. O vetor de mutação resultante é cruzado com o vetor pai correspondente para gerar um vetor experimental ou descendente. Então, em um processo de seleção um-para-um de cada par de progenitores e vetores pai, aquele com um melhor valor de aptidão sobrevive e entra na próxima geração. Este procedimento repete-se para cada vetor pai e os sobreviventes de todos os pares progenitores-descendentes tornam-se os pais de uma nova geração no ciclo evolutivo de busca. A pesquisa evolutiva pára quando o algoritmo converge para o ótimo ou um determinado critério de término como o número de gerações seja atingido (ZHANG; SANDERSON, 2009a).

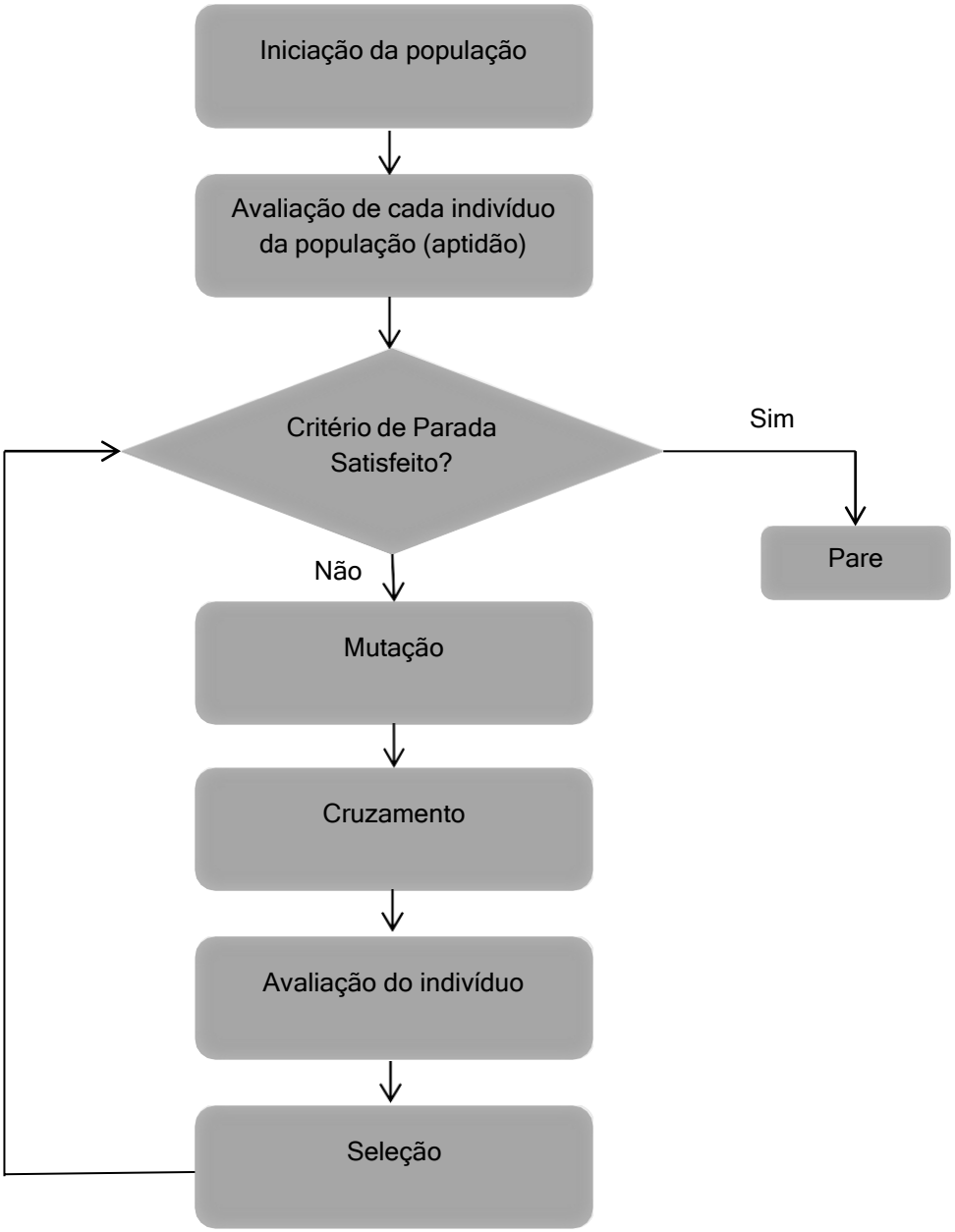
Na próxima seção, cada uma das etapas da versão clássica do algoritmo ED é apresentada em detalhes.

3.2 ALGORITMO CLÁSSICO PARA OTIMIZAÇÃO CONTÍNUA

A versão original do algoritmo de ED, também conhecida como a versão clássica, desenvolvida e apresentada por Storn e Price (1995) e Storn e Price (1997), consiste de quatro fases: inicialização; mutação; cruzamento; e seleção. No algoritmo de ED as operações de mutação, cruzamento e seleção são repetidas geração após geração até que algum critério de parada seja satisfeito (STORN; PRICE, 1997; QIN; SUGANTHAN, 2005; CHAKRABORTY; 2008; BROWNLEE, 2011).

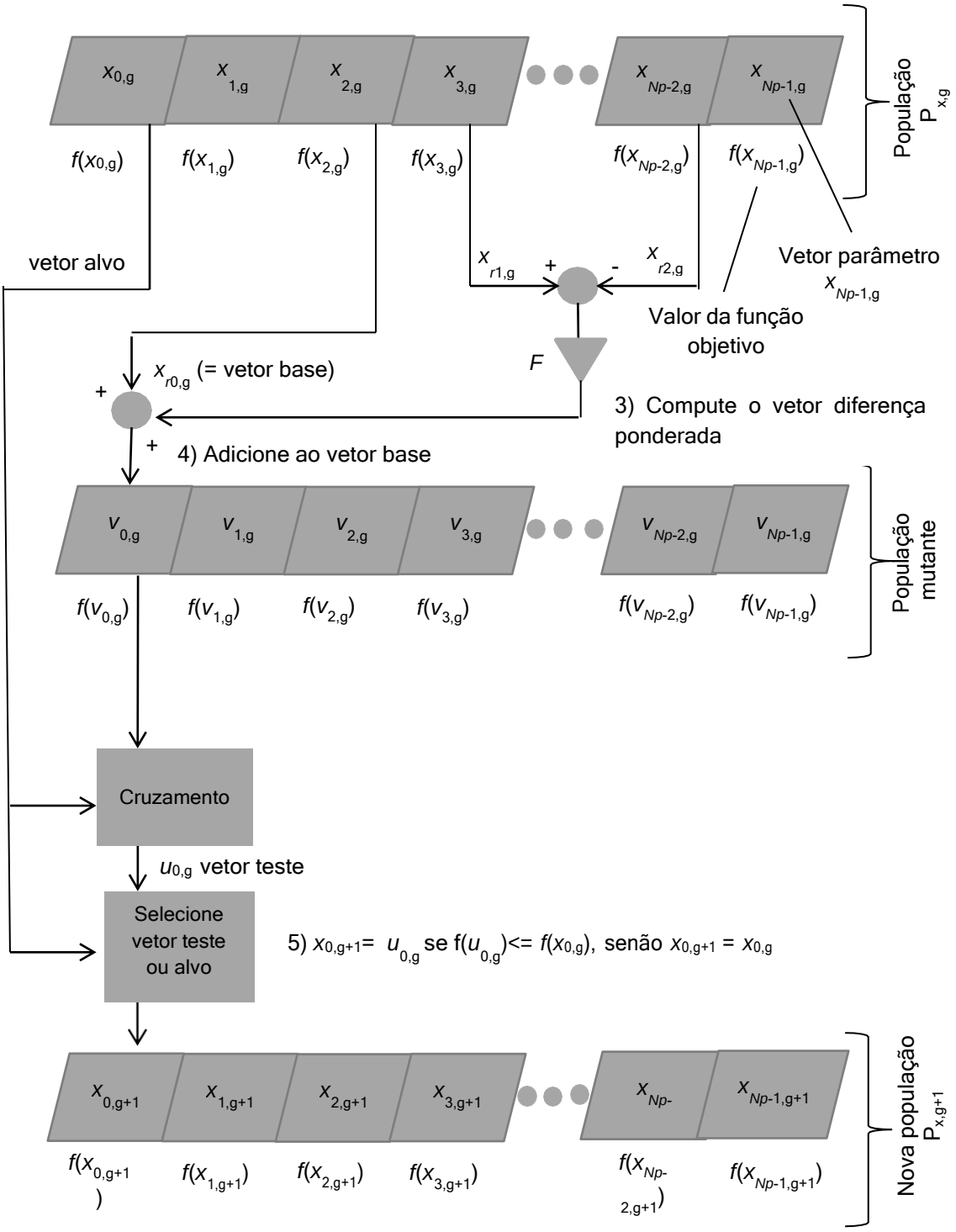
As principais etapas de um algoritmo de ED Clássico são mostradas na Figura 3.1, e um fluxograma de um algoritmo de ED Clássico é apresentado na Figura 3.2.

Figura 3.1 – Etapas de um Algoritmo de Evolução Diferencial Clássico



Fonte: Adaptado de Rosário (2011)

Figura 3.2 - Fluxograma de um Algoritmo de Evolução Diferencial Clássico



Fonte: Adaptado de Storn In Chakraborty (2008)

As etapas de um algoritmo de ED Clássico, bem como os elementos ilustrados nas Figuras 3.1 e 3.2 são apresentados com mais detalhes nas próximas subseções.

3.2.1 Estrutura Populacional

Em um algoritmo de ED, o *i*-ésimo indivíduo da população pertencente à geração atual *G* pode ser identificado como $x_{i,G}$. A população corrente, simbolizada por P_x, é composta pelos vetores $x_{i,G}$ que já tenham sido encontrados como aceitáveis, quer como pontos iniciais, quer por comparação com outros vetores (PRICE; STORN; LAMPINEN, 2005).

No algoritmo de ED $x_{i,G}$ e P_x podem ser representados como:

$$P_{x,G} = (x_{i,G}), i = 0,1,..., Np-1,\ G = 0,1,..., G_{max} \quad (3.1)$$

$$x_{i,G} = (x_{j,i,G}),\ j = 0,\ 1,..., D-1 \quad (3.2)$$

O índice $G = 0,1,..., G_{max}$ indica a geração a qual o vetor pertence. Em adição, cada vetor é designado a uma população de índice *i*, que vai de 0 a N_p. Os parâmetros dentro dos vetores são indexados com $j = 0, 1,..., D-1$ (PRICE; STORN; LAMPINEN, 2005).

Uma vez iniciado, o algoritmo de ED muta os vetores escolhidos aleatoriamente para produzir uma população intermediária, $P_{v,G}$, de N_p vetores mutantes, $v_{i,G}$ (PRICE; STORN; LAMPINEN, 2005).

$$P_{v,G} = (v_{i,G}), i = 0,1,..., Np-1,\ G = 0,1,..., G_{max} \quad (3.3)$$

$$v_{i,G} = (v_{j,i,G}),\ j = 0,1,..., D-1 \quad (3.4)$$

Cada vetor na população corrente é então recombinado com um mutante para produzir uma população, P_u, de N_p vetores experimentais, $u_{i,G}$ tal que:

$$P_{u,G} = (u_{i,G}), i = 0,1,..., Np-1,\ G = 0,1,..., G_{max} \quad (3.5)$$

$$u_{i,G} = (u_{j,i,G}),\ j = 0,1,..., D-1 \quad (3.6)$$

Durante o cruzamento, vetores experimentais substituem a população mutante, então uma única matriz pode conter ambas as populações (PRICE; STORN; LAMPINEN, 2005).

3.2.2 Geração da População Inicial

Antes de a população poder ser inicializada, os limites superiores e inferiores para cada parâmetro devem ser especificados. Estes valores podem ser coletados em dois vetores de inicialização x^L e x^U para os quais sobrescritos *L* e *U* indicam os limites inferior e superior, respectivamente.

Uma vez que os limites de inicialização foram especificados, um gerador de números aleatórios atribui a cada parâmetro de cada vetor um valor dentro do intervalo prescrito.

A população inicial ($G=0$) de vetores $x_{i,0}, i = 1,2,N_p$ é formada aleatoriamente e uniformemente distribuída entre os limites superiores e inferiores do problema e deve cobrir a totalidade do espaço de busca (STORN; PRICE, 1997; QIN; SUGANTHAN, 2005). Deste modo, o valor inicial ($G=0$) do *j*-ésimo componente do *i*-ésimo vetor é:

$$x_{j,i,0} = rand_{j,}(0,1).(b_{j,U} - b_{j,L}) + b_{j,L} \tag{3.7}$$

O gerador de números aleatórios, $rand_{j,i}(0,1)$ retorna um número aleatório uniformemente distribuído no intervalo [0,1), ou seja, $0 \leq rand_j(0,1) < 1$. O índice *j* em $rand_j$ indica que um novo valor aleatório é gerado para cada parâmetro. Mesmo se uma variável for discreta ou inteira, ela deve ser inicializada com um valor real, uma vez que o algoritmo de ED trata internamente todas as variáveis como valores de ponto flutuante independentemente do seu tipo (PRICE; STORN; LAMPINEN, 2005).

Como regra geral, deve-se considerar uma distribuição de probabilidade uniforme para todas as decisões aleatórias, salvo indicação contrária. No caso de uma solução preliminar estar disponível, a população inicial pode ser gerada pela adição de desvios aleatórios normalmente distribuídos à solução nominal $x_{nom,0}$ (STORN; PRICE, 1997).

Tasgetiren et al. (2011) ressaltam que o tamanho da população é uma escolha crítica para o desempenho do algoritmo de ED, porém, de acordo com Das e Suganthan (2012) sua influencia sobre o desempenho do algoritmo de ED ainda não foi extensivamente estudada e totalmente compreendida.

Storn e Price (1997) indicaram que um valor razoável para *Np* poderia ser escolhido entre 5*D* e 10*D* (sendo D a dimensionalidade do problema), enquanto Gämperle, Muller e Koumoutsakos (2002) propõem que *Np* seja escolhido entre 3*D* e 8*D* e Rönkkönen, Kukkonen e Price (2005) sugerem que *Np* seja escolhido entre 2*D* e 40*D*.

Aumentando-se o tamanho da população, aumenta-se a diversidade do universo de potenciais vetores experimentais e minimiza-se, portanto, o risco de estagnação devido a uma convergência prematura. Por outro lado, grandes valores de *Np* aumentam o tempo computacional. Portanto a escolha de *Np* é um compromisso entre tempo de cálculo e convergência (RÖNKKÖNEN, KUKKONEN e PRICE, 2005).

3.2.3 Mutação

A ED baseia-se na ideia de tomar a diferença entre dois indivíduos e adicionar uma versão escalada do vetor diferença a um terceiro indivíduo para criar uma nova solução candidata (SIMON, 2013), ou seja, o algoritmo de ED gera novos vetores parâmetros por meio da adição da diferença ponderada entre dois vetores da população a um terceiro vetor. Esta operação é denominada mutação diferencial (STORN; PRICE, 1997).

Para cada vetor alvo $x_{i,G}, i = 0,1,.., N_p - 1$ em uma geração *G*, um vetor mutante $v_{i,G}$, resultado da combinação de três diferentes vetores, escolhidos aleatoriamente, é gerado de acordo com:

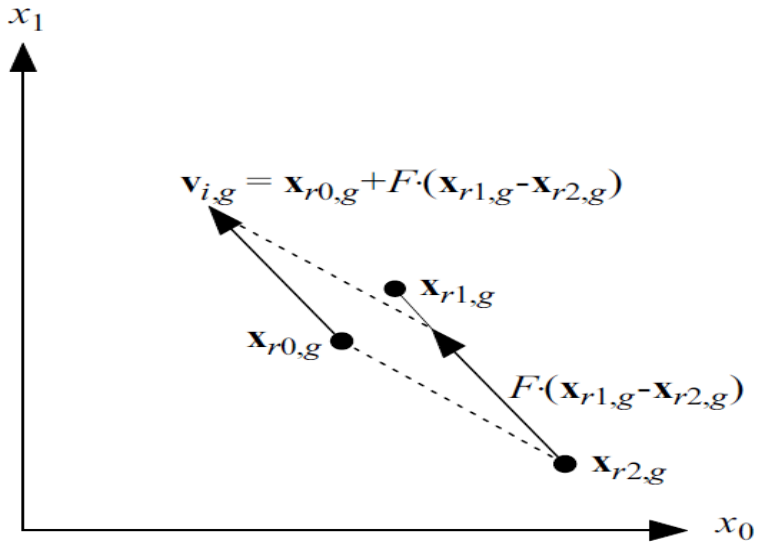
$$v_{i,G} = x_{r0,G} + F(x_{r1,G} - x_{r2,G}) \tag{3.8}$$

onde $F > 0$ e r_0, r_1, r_2 são índices aleatórios $\in \{1,2,..., Np\}$ e mutuamente diferentes. Como os inteiros escolhidos aleatoriamente r_0, r_1, r_2 também devem ser diferentes do índice de execução *i*, *Np* deve ser maior ou igual a quatro para permitir esta

condição. *F* é um fator constante e real $\in[0,2]$ o qual controla a amplificação da variação diferencial $(x_{r1,G} - x_{r2,G})$ (STORN; PRICE, 1997).

O processo de mutação diferencial, em um espaço bidimensional, onde a diferença ponderada $F(x_{r1,G} - x_{r2,G})$ é adicionada ao vetor base $x_{r0,G}$ para produzir um mutante $v_{i,G}$ é ilustrado na Figura 3.3 para duas dimensões.

Figura 3.3 - Mutação Diferencial



Fonte: Price, Storn e Lampinen (2005)

A literatura apresenta diversas formas de definir o parâmetro *F*, não havendo consenso sobre tal questão (DAS; SUGANTHAN, 2011), sendo sugerido por Simon (2013) o intervalo $0,4 < F < 0,9$ e por Das e Suganthan (2011) o intervalo $0,4 < F < 1$. De acordo com Storn e Price (1997) iniciar com *F* = 0,5 é geralmente uma escolha adequada e a utilização de valores menores que 0,4 e maiores que 1 pode ocasionar uma degradação do desempenho do algoritmo. De acordo com Rönkkönen, Kukkonen e Price (2005), típicos valores para o fator de mutação *F* seriam 0,4 < *F* < 0,95 sendo *F* = 0,9 um bom compromisso entre velocidade e probabilidade de convergência. Todavia, Gamperle, Muller e Koumoutsakos (2002) consideram *F* = 0,6 uma adequada escolha inicial.

Gamperle, Muller e Koumoutsakos (2002) destacam que embora valores elevados de *F* aumentem a probabilidade de o algoritmo escapar de um mínimo local, valores *F > 1* diminuem consideravelmente a velocidade de convergência.

Price, Storn e Lampinen (2005) estabelecem que o fator de mutação *F* deve possuir um limite superior e inferior, e ressaltam que para evitar uma convergência prematura, *F* deve ser suficientemente grande para neutralizar o efeito da seleção. Além disto, valores pequenos de *F* reduzem a chance de se escapar de um mínimo local.

3.2.4 Cruzamento

Após a fase de mutação uma operação de cruzamento é introduzida para aumentar a diversidade de vetores parâmetros perturbados (STORN; PRICE, 1997). A ED emprega cruzamento uniforme, também referenciado como cruzamento discreta, que constrói vetores de teste de valores de parâmetros que foram copiados a partir de dois vetores diferentes (PRICE; STORN; LAMPINEN, 2005).

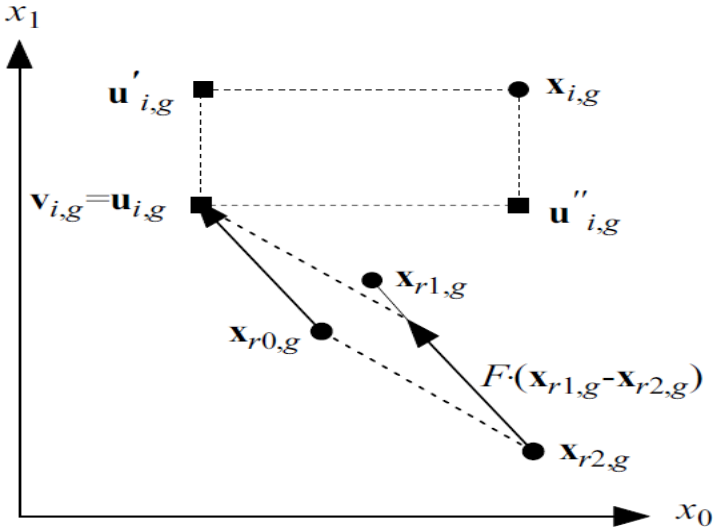
O algoritmo de ED cruza cada vetor com um vetor mutante, conforme segue:

$$u_{i,G} = u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{se}(rand_{i,j}[0,1] \leq CR) \text{ ou } j = j_{rand} \\ x_{j,i,G} & \text{caso contrário} \end{cases} \quad (3.9)$$

onde $rand_{i,j}[0,1]$ é um número aleatório gerado com distribuição uniforme, o qual é chamado um novo para cada *j-ésima* componente do *i-ésimo* vetor parâmetro. O *CR* é uma constante de cruzamento, tal que $CR \in [0,1]$ especificada pelo usuário. $j_{rand} \in 1,2,...,D$ é um índice escolhido aleatoriamente, o qual garante que $u_{i,G}$ receba pelo menos um parâmetro de $v_{i,G}$ (STORN; PRICE, 1997; DAS; SUGANTHAN, 2011). De acordo com Das e Suganthan (2012) quanto menor o valor de *CR*, menor a probabilidade de o vetor experimental receber componentes do vetor doador.

A Figura 3.4 ilustra os possíveis vetores experimentais que podem resultar do cruzamento uniformemente um vetor mutante $v_{i,G}$, com o vetor $x_{i,G}$ em duas dimensões.

Figura 3.4 - Possíveis Vetores Experimentais Adicionais Resultantes do Cruzamento Uniforme de $x_{i,G}$ e $v_{i,G}$



Fonte: Price, Storn e Lampinen (2005)

De acordo com Tasgetiren et al. (2011) *CR* é geralmente relacionado à velocidade de convergência. Para Gamperle, Muller e Koumoutsakos (2002) valores elevados para *CR* podem levar a convergência prematura ou a diminuição da velocidade de convergência, devendo ser escolhido um valor entre 0,3 e 0,9 para a taxa de cruzamento. Rönkkönen, Kukkonen e Price (2005) sugerem que a taxa de cruzamento deve ser escolhida $0 \leq CR \leq 0,2$ para funções separáveis ou $0,9 \leq CR \leq 1$ para funções não separáveis e multimodais.

3.2.5 Seleção

Para decidir se deve ou não tornar-se um membro da geração *G+1*, o vetor de ensaio $u_{i,G}$ é comparado com o vetor alvo $x_{i,G}$ usando um critério ganancioso (STORN; PRICE, 1997).

Se o vetor experimental produzir um valor menor do que a função de custo do vetor alvo, o vetor de ensaio substitui o vetor alvo na geração seguinte. Esta última operação é denominada de seleção. Cada vetor da população, a cada geração tem

que servir uma vez como o vetor alvo para que as *Np* competições aconteçam na geração (STORN; PRICE, 1997).

O valor de aptidão de cada vetor teste é comparado ao valor de aptidão de seu vetor alvo correspondente na população corrente. Se o vetor teste tiver valor de aptidão menor ou igual (para problemas de minimização) que o vetor alvo correspondente, o vetor teste substituirá o vetor alvo e entrará na população da próxima geração. Caso contrário, o vetor alvo permanece na população para a próxima geração (QIN; SUGANTHAN, 2005). Ou seja, se o vetor $u_{i,G}$ produz um valor menor do que a função de custo de $x_{i,G}$, então $x_{i,G+1}$ é definido como $u_{i,G+1}$, caso contrário, o antigo valor de $x_{i,G}$ é retido (STORN; PRICE, 1997), conforme equação (3.10) a seguir.

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{se } f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} & \text{casocontrário} \end{cases} \tag{3.10}$$

Uma vez que a nova população é gerada, o processo de mutação, cruzamento e seleção é repetido até que o ótimo seja obtido, ou uma condição de término seja satisfeita, por exemplo, o número de gerações atinge um máximo pré-estabelecido, G_{max}.

3.2.6 Estratégias de Evolução Diferencial

O esquema de mutação mostrado na seção 3.1.3, não é a única variante da ED que provou ser útil (STORN; PRICE, 1997). O vetor mutante pode geralmente ser gerado usando cinco diferentes esquemas de mutação e cruzamento, propostos por Storn e Price (1995,1997), que dão origem a dez variantes de ED, conhecidas na literatura como Estratégias de Evolução.

De acordo com Davendra e Onwubolu (2007) cada estratégia é dependente de três fatores:

i. A solução a ser perturbada;
ii. O número de diferentes soluções consideradas para a perturbação; e
iii. O tipo de cruzamento utilizado.

As estratégias de ED são representadas como $ED/x/y/z$, sendo que:

- x determina o vetor que será perturbado, podendo ser *"rand"* que significa que um vetor da população foi escolhido aleatoriamente ou *"best"* que significa o vetor de menor custo da população foi escolhido;
- y determina a quantidade de diferenças ponderadas utilizadas para a perturbação de *x*, podendo ser *"1"* ou *"2"*; e
- z especifica qual o tipo de cruzamento, podendo ser *"exp"* que significa exponencial ou *"bin"* que significa binomial.

A principal diferença entre os dois tipos de cruzamento é que no cruzamento binomial, as escolhas dos componentes que serão herdados do pai são independentes, enquanto no cruzamento exponecial é selecionada uma subsequência contínua de componentes (CZAIKOSKI et al., 2016).

As dez estratégias de ED, desenvolvidas por Storn e Price (1995, 1997) que podem ser adotadas no desenvolvidos de algoritmos de ED são:

1. ED/rand/1/bin : $v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G})$
2. ED/rand/1/exp : $v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G})$
3. ED/best/1/bin : $v_{i,G} = x_{best,G} + F.(x_{r0,G} - x_{r1,G})$
4. ED/best/1/exp : $v_{i,G} = x_{best,G} + F.(x_{r0,G} - x_{r1,G})$
5. ED/rand − to − best/1/bin: $v_{i,G} = x_{i,G} + F.(x_{best,G} - x_{i,G}) + F.(x_{r0,G} - x_{r1,G})$ [1]
6. ED/rand − to − best/1/exp : $v_{i,G} = x_{i,G} + F.(x_{best,G} - x_{i,G}) + F.(x_{r0,G} - x_{r1,G})$
7. ED/best/2/bin : $v_{i,G} = x_{best,G} + F.(x_{r0,G} - x_{r1,G}) + F.(x_{r2,G} - x_{r3,G})$
8. ED/best/2/exp : $v_{i,G} = x_{best,G} + F.(x_{r0,G} - x_{r1,G}) + F.(x_{r2,G} - x_{r3,G})$
9. ED/rand/2/bin : $v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G})$
10. ED/rand/2/exp : $v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G})$

A versão clássica do algoritmo ED apresentada na seção 3.1 é definida como *ED/rand/1/bin*, pois o vetor base é escolhido aleatoriamente com distribuição uniforme (*x = rand*), apenas 1 vetor diferença é utilizado para perturbar o vetor base (*y = 1*) e o cruzamento é do tipo binomial (*z = bin*) (STORN; PRICE, 1997).

[1] A estratégia *DE/rand-to-best* também é conhecida na literatura como *DE/target-to-best* (PRICE, STORN e LAMPINEN, 2005).

Ressalta-se que a estratégia de ED adotada no desenvolvimento dos métodos de solução é uma questão muito importante, pois o algoritmo pode apresentar diferentes comportamentos baseados na estratégia selecionada.

3.2.7 Pseudocódigo de um Algoritmo de Evolução Diferencial

A seguir um pseudocódigo para um algoritmo de Evolução Diferencial com a utilização da estratégia ED/rand/1/bin: $v_{i,G+1} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G})$ é apresentado no Quadro 3.1, lembrando que *Np* indica o tamanho da população e *D* indica o número de dimensões do problema. O índice $G = 0,1,..., G_{max}$ indica a geração a qual um vetor pertence. Cada vetor é designado a uma população de índice *i*, que vai de 0 a $N_p - 1$. Os parâmetros dentro dos vetores são indexados com $j = 0, 1,..., D - 1$.

Quadro 3.1- Pseudocódigo de um Algoritmo de Evolução Diferencial Clássico

Pseudocódigo de um Algoritmo de Evolução Diferencial Clássico

Ler os valores dos parâmetros de controle do ED: Taxa de Mutação (*F*), Taxa de Cruzamento (*CR*) e tamanho da população (*Np*)

$G \leftarrow 0$

Inicializar aleatoriamente uma população $P_{x,G} = \{x_{i,G};\ i = 0,1,...,Np-1\}$ com cada indivíduo uniformemente distribuído entre os limitantes

Enquanto condição de término não satisfeito faça

  Para $i = 0,1,...,Np-1$ faça para cada individuo sequencialmente

    Selecionar aleatoriamente $r_0, r_1, r_3 \in \{0,1,...,Np-1\}$ com distribuição uniforme

    Selecionar aleatoriamente $j_{rand} \in \{1,2,...D\}$ com distribuição uniforme

    Para $j = 1$ até D faça

      Se $rand_{i,j}(0,1] \leq CR \vee j = j_{rand}$

        $u_{i,G} = v_{i,G} = x_{r0,G} + F(x_{r1,G} - x_{r2,G})$

      Senão $u_{i,G} = v_{i,G} = x_{i,G}$

      Fim //Se

    Fim //Para

  Fim //Para

  Para $i = 0,1,...,Np-1$

    Se $f(u_{i,G}) \leq f(x_{i,G})$ então

      $x_{i,G+1} \leftarrow u_{i,G}$

    Senão $x_{i,G+1} \leftarrow x_{i,G}$

    Fim //Se

  Fim //Para

  $G \leftarrow G+1$

Fim //Enquanto

Fonte: Adaptado de Gaspar-Cunha, Antunes e Takahashi (2012) e Das e Suganthan (2011)

3.3 EVOLUÇÃO DIFERENCIAL ADAPTATIVA/AUTOADAPTATIVA

A ED tem se mostrado ser um algoritmo evolutivo simples, mas poderoso para otimização global para muitos problemas do mundo real. No entanto, como outros algoritmos evolutivos, o desempenho da ED é dependente da configuração dos parâmetros de controle, tais como o fator de mutação, a probabilidade de cruzamento e o tamanho da população (ZHANG; SANDERSON, 2009b).

Embora já sejam sugeridos valores destes parâmetros na literatura especializada (STORN; PRICE, 1995, 1997; GAMPERLE; MULLER; KOUMOUTSAKOS, 2002; RÖNKKÖNEN; KUKKONEN; PRICE, 2005; DAS; SUGANTHAN, 2011; SIMON, 2013), a interação entre a parametrização e o desempenho de convergência ainda é complicado e não totalmente esclarecido. Isto deve-se principalmente porque não existe um valor único parâmetro que é adequado para vários problemas ou executa consistentemente bem, mesmo em diferentes estágios de evolução de um único problema (ZHANG; SANDERSON, 2009b).

O método de tentativa e erro para ajustar os parâmetros de controle geralmente requer extensivos ensaios de otimização, mesmo para um algoritmo (por exemplo, o clássico *ED/rand/1/bin*) que fixa os parâmetros em toda a pesquisa evolucionária. Com base nesta consideração, diferentes mecanismos para atualizar dinamicamente os parâmetros de controle sem o conhecimento prévio do usuário do problema ou a interação durante o processo de pesquisa tem sido introduzidos. Além disso, os parâmetros de controle, se bem concebidos, são capazes de melhorar o desempenho de convergência no algoritmo (ZHANG; SANDERSON, 2009b).

Alguns exemplos de mecanismos adaptativos e/ou autoadaptativos de atualização dos parâmetros do algoritmo de ED podem ser verificados em Abbass (2002), Liu e Lampinen (2002), Qin e Suganthan (2005), Liu e Lampinen (2005), Brest et al. (2006), Teo (2006), Brest et al. (2007), Noman e Iba (2008), Yang, Tang e Yao (2008), Das et al. (2009), Qin, Huang e Suganthan (2009), Zhang e Sanderson (2009a), Wang, Cai e Zhang (2011), Mallipeddi et al. (2011), Tanabe e Fukunaga (2013) e Tanabe e Fukunaga (2014).

Na seção 3.3.1 são explicitadas as diferentes classes de mecanismos de ajustes ou adaptação de parâmetros de controle dos algoritmos AEs, e consequentemente dos algoritmos de ED.

3.3.1 Mecanismos de Adaptação de Parâmetros de Controle

Os mecanismos de adaptação dos AEs propostos na literatura podem ser categorizados com base nas mudanças dos parâmetros de controle. De acordo com Angeline (1995), Eiben, Hinterding e Michalewicz (1999), Zhang e Sanderson (2009b) e Eiben e Smith (2015) são verificadas três classes de mecanismos de ajuste de parâmetros de controle, conforme segue:

i. Mecanismos determinísticos: O parâmetro de controle é alterado por uma regra determinística sem levar em conta qualquer *feedback* a partir da busca evolutiva. O algoritmo de Evolução Diferencial Clássico proposto por Storn e Price (1995, 1997) enquadra-se nesta categoria;
ii. Mecanismos adaptativos: O *feedback* da pesquisa evolutiva é usado para alterar dinamicamente os parâmetros de controle. Nesta classe se enquadram os trabalhos de Liu e Lampinen (2005) e Qin e Suganthan (2005); e
iii. Mecanismos autoadaptativos: Um método de evolução é utilizado para conduzir a autoadaptação dos parâmetros de controle. Os parâmetros estão diretamente associados com os indivíduos e, portanto, eles próprios sofrem mutação e cruzamento. Uma vez que os melhores valores dos parâmetros tendem a gerar indivíduos que são mais propensos a sobreviver, estes valores podem ser propagados para mais descendentes. Algoritmos de ED propostos por Abbass (2002) e Teo (2006) pertencem a esta categoria.

Zang e Sanderson (2009b) mencionam que quando comparado com o método determinístico, os métodos adaptativos e autoadaptativos, se bem concebidos, podem aumentar a robustez de um algoritmo, além de melhorar a taxa de convergência por meio da adaptação dos parâmetros de controle, uma vez que os valores são adequados em diferentes fases de evolução de um problema de optimização.

Ao longo dos últimos anos, inúmeras variantes do algoritmo ED foram desenvolvidas com o propósito de melhorar o desempenho no processo de otimização. Algumas versões do algoritmo ED com parâmetros de controle adaptativos são apresentadas na seção 3.3.2.

3.3.2 Algoritmos de Evolução Diferencial Adaptativos/Autoadaptativos

Algumas versões do algoritmo de ED com parâmetros de controle adaptativos/autoadaptativos para a solução de problemas de otimização mono-objetivo destacadas na literatura especializada (DAS; SUGANTHAN, 2011; SUGANTHAN, 2012; DAS; MULLICK; SUGANTHAN, 2016) são:

- jDE (*j Differential* Evolution), proposto por Brest et al. (2006);
- DEahcSPX (*Differential Evolution using Adaptive Hill-Climbing Crossover-Based Local Search and Simplex Crossover*), proposto por Noman e Iba (2008);
- SaDE (*Self-adaptive Differential Evolution*) proposto por Qin e Suganthan (2005) e estendido por Qin, Huang e Suganthan (2009);
- JADE (*J Adaptive Differential Evolution*) prospoto por Zhang e Sanderson (2007) e Zhang e Sanderson (2009a);
- DEGL (*Differential Evolution with Global and Local Neighborhoods*) proposto por Das et al. (2009);
- CoDE (*Composite Differential Evolution*) proposto por Wang, Cai e Zhang (2011);
- EPSDE (*Ensemble of Parameters and mutation Strategies in Differential Evolution*) proposto por Mallipeddi et al. (2011);
- SHADE (*Success-History based Adaptive Differential Evolution*) desenvolvido por Tanabe e Fukunaga (2013);
- L-SHADE (*Success-History Based Adaptive Differential Evolution With Linear Population Size Reduction*) proposto por Tanabe e Fukunaga (2014); e
- SPS-L-SHADE-EIG (*Success-History Based Adaptive Differential Evolution with Linear Population Size Reduction with an Eigenvector-Based Crossover and a successful-parent selecting*) proposto por Guo et al. (2015).

O Quadro 3.1 a seguir sumariza as características principais dos algoritmos com parâmetros adaptativos destacados na literatura especializada.

Quadro 3.2 - Principais características dos algoritmos com parâmetros adaptativos (continua...)

Algoritmo	**Características Principais**
jDE	Neste algoritmo, os parâmetros *F* e *CR* são configurados autoadaptativamente durante o processo de otimização. Os parâmetros de controle, F_i e CR_i , são aplicados em nível de indivíduo $x_{i,G}$ e ajustados por meio de evolução. Os parâmetros F_i e CR_i, são atualizados durante o processo de otimização de acordo com duas probabilidades r_1 e r_2. No início do processo de otimização F_i e CR_i são inicializados a 0,5 e 0,9, respectivamente.
DEahcSPx	Neste algoritmo, o algoritmo clássico de ED é combinado com uma busca local denominada *Adaptive Hill-Climbing Crossover-Based Local Search* (AHCXLS). A busca local AHCXLS incorporada ao DE determina adaptativamente o comprimento da busca, tendo *feedback* da busca. No algoritmo DEahcSPX, a cada geração *G*, antes de se executar as operações de mutação, cruzamento e seleção do algoritmo ED, a busca local AHCXLS é aplicada.
SaDE	Neste algoritmo a estratégia de ED e os parâmetros de controle *F* e *CR* são gradualmente autoadaptados de acordo com a experiência de aprendizagem. No algoritmo SaDE as estratégias de ED são escolhidas para ser aplicada a indivíduos na população corrente com probabilidades proporcionais às suas taxas de sucesso anteriores para gerar potencialmente boas novas soluções. A cada geração *G*, um fator de mutação e uma taxa de cruzamento são gerados aleatoriamente para cada vetor alvo a partir de uma distribuição normal. F_i é gerado a partir de uma distribuição normal com média de 0,5 e desvio padrão de 0,3, enquanto CR_i é gerado a partir de uma distribuição normal com média CRm_k e desvio padrão de 0,1.
JADE	Neste algoritmo uma nova estratégia de mutação e um novo processo de autoadaptação dos parâmetros *F* e *CR* são implementados. Este novo processo de autoadaptação dos parâmetros de controle atualiza dinamicamente o fator de mutação e a taxa de cruzamento durante o processo de otimização. Em cada geração *G,* uma probabilidade de cruzamento CR_i para cada individuo $x_{i,\,G}$ é independentemente gerado, de acordo com uma distribuição normal, com média μ_{CR} e desvio padrão 0,1, respectivamente, e então truncado para [0, 1]. Também em cada geração *G,* o fator de mutação F_i de cada indivíduo $x_{i,\,G}$, é independentemente gerado, de acordo com uma distribuição de Cauchy, com parâmetro de locação μ_F e parâmetro de escala de 0,1.

Fonte: o autor, 2017.

Quadro 3.2 - Principais características dos algoritmos com parâmetros adaptativos/autoadaptativos (continua...)

Algoritmo	**Características Principais**
DEGL	No algoritmo DEGL é desenvolvida uma estratégia de mutação que utiliza um modelo de vizinhança local e um modelo de vizinhança global, responsáveis por gerar os vetores doador local e doador global, respectivamente. Neste algoritmo o fator de peso para combinar os dois modelos de vizinhança é determinado por meio de cinco esquemas, conforme segue: fator de ponderação aumentado; incremento linear; incremento exponencial; fator de ponderação aleatório; e fator de ponderação autoadaptativo.
CoDE	Este algoritmo utiliza três estratégias de geração do vetor experimental e três configurações para os parâmetros de controle F e CR . No algoritmo CoDE, a cada geração e para cada vetor alvo, as três estratégias são utilizadas, sendo cada uma associada a uma combinação de valores dos parâmetros de controle selecionada aleatoriamente a partir do conjunto de combinações de valores pré definidos, para gerar um novo vetor experimental.
EPSDE	Este algoritmo emprega um conjunto pré-definido de estratégias de geração do vetor experimental e conjuntos de valores pré-definidos para os parâmetros de controle para gerar os vetores experimentais. No EPSDE um conjunto de estratégias de mutação distintas, juntamente com um conjunto de valores para cada parâmetro de controle coexiste em todo o processo de evolução e concorre para produzir uma prole.
SHADE	Este algoritmo utiliza um esquema de adaptação dos parâmetros de controle baseado em histórico. O algoritmo SHADE mantém uma memória histórica com *H* entradas para os parâmetros de controle do *ED*, F e CR , M_{CR} e M_F . Os conteúdos de $M_{CR,i}$, $M_{F,i}$ $(i=1,2,...,H)$ são todos inicializado em 0,5.
L-SHADE	Éste algoritmo constitui uma ampliação do algoritmo SHADE, em que o algoritmo LPSR (*Linear Population Size Reduction*), um método determinístico simples, é incorporado com o objetivo de reduzir continuamente o tamanho da população de acordo com uma função linear. Neste algoritmo a população inicial P_0 de tamanho *Np* é inicializada aleatoriamente e, assim como na L-SHADE o tamanho da população na geração 1 é N^{init} e a população no final da execução é $N^{\min}$.

Fonte: o autor, 2017.

Quadro 3.2 - Principais características dos algoritmos com parâmetros adaptativos (...fim)

Algoritmo	Características Principais
SPS-L-SHADE-EIG	É uma ampliação do algoritmo L-SHADE, que incorpora um cruzamento baseado em autovetor, denominado EIG (*Eigenvector-Based*) e um procedimento para a seleção de pais, denominada SPS (*Successful-Parent Selecting*). Como no algoritmo SPS-L-SHADE-EIG, a população inicial P_0 de tamanho *Np* é inicializada aleatoriamente e, assim como na L-SHADE o tamanho da população na geração 1 é N^{init} e a população no final da execução é $N^{\min}$. Neste algoritmo o procedimento SPS, que visa fornecer soluções mais promissores e ajudar o ED a escapar de situação de estagnação, tem como ideia básica selecionar pais das soluções mais recentemente atualizadas quando a estagnação é detectada.

Fonte: o autor, 2017.

3.3 EVOLUÇÃO DIFERENCIAL DISCRETA

Após a introdução do algoritmo de Evolução Diferencial Clássico, muitas variações foram introduzidas e após ser provado o sucesso da ED para problemas com domínios contínuos, os pesquisadores estenderam a ED para problemas com domínios discretos (SIMON, 2013). Dado que este algoritmo de ED foi originalmente projetado para problemas com domínios contínuos, este não pode ser aplicado a problemas discretos ou de permutação sem modificações (ONWUBOLU; DAVENDRA, 2009).

De acordo com Simon (2013) a única fase em que os domínios discretos causam problemas em um algoritmo de ED é na geração do vetor mutante. De acordo com Onwubolu e Davendra (2009) o mecanismo interno de cruzamento e mutação do ED invariavelmente altera qualquer valor aplicado a um número real, o que por si só poderá levar a soluções inviáveis. O objetivo passa então a ser a transformação da população ou do mecanismo interno de cruzamento e mutação do algoritmo de ED.

Vários pesquisadores decidiram não modificar de forma alguma a operação das estratégias de ED, mas manipular a população de tal forma que habilite o ED a operar sem impedimentos. Uma vez que a solução para uma população é permutativa, rotinas de conversão adequadas são necessárias para alterar a solução

de inteiro para real e, em seguida, volta para inteiro após o cruzamento (ONWUBOLU; DAVENDRA, 2009).

A estratégia mais simples para uso do algoritmo de ED para a solução de problemas de otimização com variáveis no domínio do conjunto dos números inteiros consiste na aproximação dos valores fracionários para o inteiro mais próximo quando necessário. No entanto, a literatura especializada contempla diversas abordagens para tratar a questão da conversão dos domínios nos algoritmos de ED, algumas das quais são apresentadas na seção 3.3.1.

3.3.1 Abordagens de Evolução Diferencial Discreta

Alguns pesquisadores vêm apresentando abordagens para lidar com o algoritmo de ED para resolver problemas de otimização combinatória baseados em permutação. Estas abordagens basicamente adaptam os operadores de mutação para que as soluções resultantes sejam permutações válidas. Algumas dessas abordagens são apresentadas nas subseções seguintes.

3.3.1.1 Abordagem por Matriz de Permutação

A abordagem por matriz de permutação, apresentada por Price, Storn e Lampinen (2005) baseia-se na ideia de encontrar uma matriz permutativa que relaciona dois vetores (ONWUBOLU; DAVENDRA, 2009).

Na abordagem por matriz de permutação as soluções são representadas por vetores de permutação de inteiros e a partir desta representação, a diferença entre duas soluções candidatas $x_{r2,G}$ e $x_{r3,G}$ é definida pela relação expressada na equação 3.11, tal que

$$x_{r2,G} = P x_{r3,G} \tag{3.11}$$

onde *P* é denominada matriz de permutação e mapeia o vetor $x_{r3,G}$ em outro vetor $x_{r2,G}$.

A matriz *P* pode ser interpretada como um conjunto de movimentos de troca que, quando aplicados em $x_{r3,G}$ se obtém $x_{r2,G}$. O operador de mutação diferencial, a partir da matriz *P* é definido por 3.1, tal que.

$$v_{i,G} = P x_{r1,G} \tag{3.12}$$

onde os movimentos de troca definidos por *P*, que levam $x_{r3,G}$ a $x_{r2,G}$ são aplicados a uma terceira solução candidata, a solução base $x_{r1,G}$.

A matriz de permutação não define o processo de cruzamento, que geralmente não é utilizada nesta abordagem. Com isso, a solução obtida pela mutação diferencial é comparada com a solução alvo $x_{i,G}$ e, a melhor solução deverá compor a população da próxima iteração.

Para ilustrar a abordagem por matriz de permutação considere os vetores $x_{r1,G}$, $x_{r2,G}$ e $x_{r3,G}$:

$$x_{r1,G} = \begin{pmatrix} 2 \\ 4 \\ 1 \\ 5 \\ 3 \end{pmatrix}, x_{r2,G} = \begin{pmatrix} 1 \\ 4 \\ 3 \\ 5 \\ 2 \end{pmatrix}, x_{r3,G} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 5 \\ 2 \end{pmatrix}$$

onde cada vetor é uma permutação válida. A matriz de permutação *P* que mapeia $x_{r3;G}$ em $x_{r2;G}$, sendo $P_{xr3;G} = x_{r2;G}$, é dada por:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

e, com isso, o novo vetor $v_{i;g}$ obtido através da adaptação da mutação diferencial é dado por:

$$v_{i,G} = P x_{r1,G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \\ 1 \\ 5 \\ 3 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

O fator de mutação *F*, similar à mutação diferencial original, pode ser usado para escalonar o efeito da matriz de permutação nas novas soluções (PRICE; STORN; LAMPINEN, 2005). Para isso, o fator de mutação $F \in [0,1]$ define a probabilidade de cada movimento ser realizado. Através da abordagem por matriz de permutação, as novas soluções obtidas serão permutações válidas.

3.3.1.2 Abordagem por Matriz de Adjacência

Em problemas onde as permutações definem ciclos, vetores como $(1,2,3,4,5)^T$ e $(3,4,5,1,2)^T$ representam a mesma solução, dado que estes vetores estão apenas rotacionados. No entanto, a abordagem por matriz de permutação não é capaz de identificar estas situações.

Devido à dificuldade em identificar vetores rotacionados através da abordagem por matriz de permutação, Price, Storn e Lampinen (2005) também apresentaram a abordagem por matriz de adjacência, onde as soluções são representadas por matrizes de adjacência, e não por vetores de permutação de inteiros.

A matriz de adjacência define o subgrafo do ciclo solução. Com esta estratégia, permutações iguais, mesmo que rotacionadas geram a mesma matriz de adjacência. Assim, a diferença entre elas sempre será zero.

Price, Storn e Lampinen (2005) definiram a notação $(x + y) \bmod 2 = x \oplus y$ que é uma abreviatura para a adição do módulo 2, também conhecida como a operação "exclusiva ou" lógica para o funcionamento das matrizes.

A matriz diferença Δ_{ij} expressada por 3.13 é definida como análoga do vetor diferença do ED tradicional.

$$\Delta_{ij} = A_i \oplus A_j \tag{3.13}$$

A matriz de diferença é definida pela equação 3.14 tal que

$$D_{i,G} = x_{r2,G} \oplus x_{r3,G} \tag{3.14}$$

Após obter a diferença entre duas matrizes de adjacência, o resultado obtido é combinado à matriz de adjacência da solução base.

Considerando como exemplo as matrizes A_1 e A_2

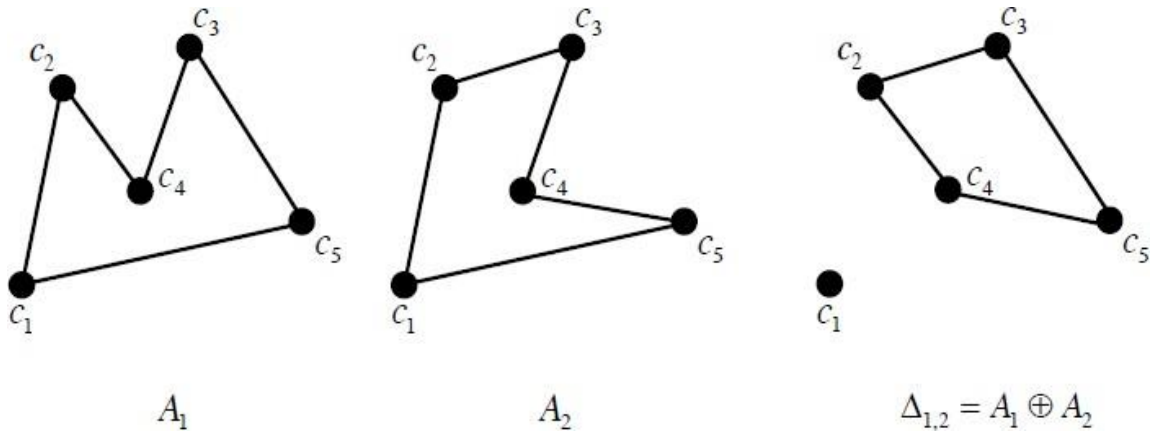
$$A_1 = \begin{pmatrix} 0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0 \end{pmatrix} \text{ e } A_2 = \begin{pmatrix} 0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0 \end{pmatrix}$$

e suas diferenças dadas como

$$\Delta_{12} = A_1 \oplus A_2 = \begin{pmatrix} 0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1 \\ 0\,0\,1\,1\,0 \end{pmatrix}$$

Da definição de A_1 há 1 na coluna 1 nas linhas {2 e 5}, na coluna 2 nas linhas {1 e 4}, na coluna 3 nas linhas {4 e 5}, na coluna 4 nas linhas {2 e 3} e na coluna 5 nas linhas {1 e 3}. Estes números emparelhados definem relacionamentos de adjacência. Considerando {2 e 5} e {4 e5} verificamos que 5 é comum e {2 e 5} adjacentes. Considerando {1 e 4} e {1 e 3} verificamos que 1 é comum e {4 e 3} adjacentes. Continuando desta maneira pode-se observar que a Figura 3.5 mostra a interpretação gráfica de A_1, A_2 e $\Delta_{i,j}$.

Figura 3.5 – Interpretações Gráficas de A_1, A_2 e da Matriz Diferença $\Delta_{i,j}$



Fonte: Onwubolu e Davendra (2009)

3.3.1.3 Abordagem de Índice de Posição Relativa

Na abordagem de Índice de Posição Relativa (RPI - *Relative Position Index*), proposta por Lichtblau (2002), permutações são obtidas por determinação dos tamanhos relativos dos diferentes parâmetros que definem uma instância (ONWUBOLU; DAVENDRA, 2009).

A ideia básica desta abordagem é de que a diferença de dois vetores, adicionada a outro vetor pode ser transferida diretamente para o campo das permutações, ou o grupo de permutações. Assim como dois vetores no espaço real definem um vetor de diferença, que é também um vetor, duas permutações definem um mapeamento, que também é uma permutação (ONWUBOLU; DAVENDRA, 2009).

Na abordagem RPI, as soluções são codificadas como vetores de permutação de inteiros, assim como é feito na abordagem por matriz de permutação. Porém os operadores de mutação e cruzamento do ED original são mantidos inalterados (LICHTBLAU, 2009).

Após a aplicação dos operadores de mutação e cruzamento, os vetores gerados provavelmente possuirão valores fracionários. Assim, antes das soluções serem avaliadas pela função objetivo, elas são transformadas em vetores de permutação de inteiros válidos.

O processo de conversão dos vetores, de acordo com a abordagem de Índice de Posição Relativa ocorre da seguinte forma: dado o vetor $u_{i,G}$, obtido após a mutação e cruzamento, e considerando que uma solução válida é uma permutação de rótulos numéricos de 1 a *n*, o rótulo 1 é alocado na mesma posição onde se encontra o menor valor em $u_{i,G}$, o rótulo 2 na posição do segundo menor valor em $u_{i,G}$, e assim por diante.

Para ilustrar esta abordagem, considere o vetor:

$$u_{i,G} = (0, -2, 6{,}5, 3, 3{,}5)^T$$

O rótulo 1 é alocado na posição de menor valor, ou seja, na segunda posição do vetor. O rótulo 2 é alocado na primeira posição, onde se encontra do segundo menor valor. Seguindo esta abordagem, o rótulo 3 é alocado na quarta posição, o

rótulo 4 na quinta posição e o rótulo 5 na terceira posição, resultando no seguinte vetor de permutação:

$$u'_{i,G} = (2,1,5,3,4)^T$$

Com o uso dos operadores de mutação e cruzamento do algoritmo de ED original, podem ocorrer situações onde o vetor $u_{i,G}$ gerado possua dois ou mais valores repetidos. Nestes casos, estas soluções devem ser descartadas ou, então, utilizar algum critério de desempate.

3.3.1.4 Abordagem de Posição de Menor Valor

A abordagem de Posição de Menor Valor (SPV - *Smallest Position Value*), proposto por Tasgetiren et al. (2009) é semelhante à abordagem de Índice de Posição relativa, divergindo apenas no processo de transformação dos vetores $u_{i,G}$.

Na abordagem SPV uma única representação de solução de uma formulação do ED contínuo é apresentada e então a regra SPV é usada para determinar as permutações (ONWUBOLU; DAVENDRA, 2009). A abordagem SPV realiza a transformação em duas etapas. Primeiro, é atribuído um rótulo a cada $u^j_{i,G}$ e, em seguida, eles são ordenados.

Para atribuir os rótulos, é utilizada a mesma estratégia da abordagem RPI, onde o rótulo 1 é atribuído ao menor valor, o rótulo 2 ao segundo menor valor, e assim por diante. No entanto, na abordagem SPV somente a parte inteira de $u^j_{i,G}$ é considerada para a atribuição dos rótulos. Após a atribuição dos rótulos, o vetor obtido é ordenado de acordo com a parte fracionária de $u^j_{i,G}$. Esta estratégia de atribuição de rótulos e ordenação após a atribuição é semelhante à estratégia utilizada definida pelo algoritmo *Random Keys*.

Para ilustrar esta abordagem, tomemos o vetor $u_{i,G} = (3,4,1,2,-4,6,2,7,1)^T$. Após a atribuição dos rótulos, é obtido o vetor $u'_{i,G} = (4,2,1,3,5)^T$. Agora, ordenando este vetor de acordo com a parte fracionária dos valores em $u_{i,G}$, é obtido o vetor de permutação de inteiros $u''_{i,G} = (1,3,5,2,4)^T$.

Assim como na abordagem RPI, podem ocorrer vetores $u_{i,G}$ com dois ou mais valores repetidos. Nestes casos, eles também devem ser descartados ou, então, utilizar algum critério de desempate.

3.3.1.5 Abordagem de Transformação para *Frente/Trás* (*Forward/Backward*)

A abordagem de transformação para frente/trás (*Forward/Backward Transformation*), proposta por Onwubolu (2001), também utiliza os operadores de mutação e cruzamento originais do algoritmo de ED, porém utilizando uma estratégia de mapeamento do espaço discreto para o contínuo e vice-versa.

De acordo com Onwubolu e Davendra (2009) na abordagem de transformação para frente/trás (*Forward/Backward*) duas etapas são verificadas:

i) Transformação para Frente (*Forward*): antes dos vetores serem submetidos aos operadores de mutação e cruzamento eles são mapeados em vetores de valores reais; e
ii) Transformação para Trás (*Backward*): Após o processo de mutação e cruzamento, os vetores gerados são mapeados de volta em vetores de permutação de números inteiros.

Na etapa de transformação para Frente o algoritmo utiliza a equação 3.15 para mapeamento.

$$\hat{x}_{i,G} = -1 + \alpha x_{i,G} \tag{3.15}$$

onde $\hat{x}_{i,G}$ é o resultado do mapeamento do vetor de permutação de inteiros $x_{i,G}$ em um vetor de números reais. O parâmetro α é um valor pequeno definido manualmente, sendo $\alpha = 0,5$ o valor sugerido pelos autores desta abordagem.

Na etapa de transformação para trás o vetor $\hat{u}_{i,G}$ obtido após as operações de mutação e cruzamento para um vetor de números inteiros é mapeado por meio da equação 3.16, onde

$$u^{j}_{i,G} = \left\lfloor \frac{1 + \hat{u}^{j}_{i,G}}{\alpha} + 0,5 \right\rfloor \tag{3.16}$$

Uma vez que a etapa de transformação para trás pode gerar vetores com valores inteiros repetidos, que representam soluções inválidas (PRICE; STORN; LAMPINEN, 2005), torna-se necessário adotar alguma estratégia para lidar com essa situação, seja por meio do uso de operadores de reparação ou simplesmente descartando tais soluções.

Capítulo 4 - REVISÃO DE LITERATURA

Este capítulo visa identificar o atual estado da arte das pesquisas voltadas a solução de Problemas de Programação da Produção (PPPs) utilizando abordagens de Evolução Diferencial (ED). A revisão de literatura aqui apresentada contempla trabalhos orientados aos PPPs em ambientes *Flow Shop* Permutacional (FSP).

4.1 EVOLUÇÃO DIFERENCIAL APLICADA A PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO EM *FLOW SHOP* PERMUTACIONAL

Para o levantamento das publicações, a partir do ano de 2004, sobre abordagens de ED aplicadas aos PPPs em FSP as bases de dados pesquisadas foram: *Scientific Electronic Library Online* (*Scielo*), Biblioteca Digital Brasileira de Teses e Dissertações (BDTD), *Emerald*, *Springer*, *IEEExplorer*, *Taylor* & *Francis* e *Science Direct*.

Também serviram de direcionadores para o levantamento das publicações, os *surveys* apresentados por Zhou et al. (2011), Das e Suganthan (2011), Tonge e Kulkarni (2012), Ramos e Tupia (2013), Yenisey e Yagmahan (2014), Allahverdi (2015), Allahverdi (2016) e Das, Mullick e Suganthan (2016).

Nas bases de dados pesquisadas, métodos de solução para PPPs em ambientes FSP baseados em abordagens de ED foram identificadas 40 publicações.

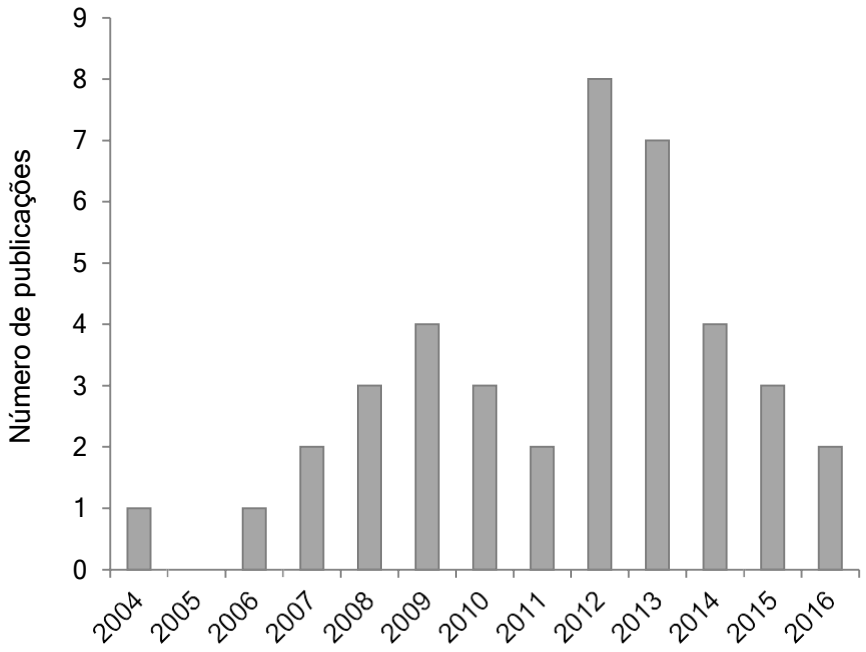
A distribuição temporal das publicações, identificadas nas bases de dados anteriormente mencionadas, é apresentada na Tabela 4.1, enquanto que a evolução do número de trabalhos publicados no decorrer dos anos é mostrada na Tabela 4.1.

Tabela 4.1 – Distribuição Temporal das Publicações Direcionadas aos Ambientes *Flow Shop* Permutacional

Ano	Número de publicações
2004	1
2005	0
2006	1
2007	2
2008	3
2009	4
2010	3
2011	2
2012	8
2013	7
2014	4
2015	3
2016	2

Fonte: o autor, 2017.

Gráfico 4.1 – Evolução das Publicações Orientadas aos Ambientes *Flow Shop* Permutacional



Fonte: o autor, 2017.

As publicações, identificadas nas bases de dados anteriormente explicitadas, que apresentam abordagens de ED aplicadas aos PPPs em ambientes FSP são relacionadas no Quadro 4.1, a seguir.

Quadro 4.1 – Publicações que Apresentam Abordagens de Evolução Diferencial Aplicadas aos Problemas de Programação da Produção em *Flow Shop* Permutacional

Ano	Referência
2004	Tasgetien et al. (2004)
2006	Qian et al. (2006)
2007	Pan, Tasgetiren e Liang (2007); Tasgetien et al. (2007)
2008	Pan e Wang (2008); Pan, Tasgetiren e Liang (2008); Qian et al. (2008)
2009	Pan, Wang e Qian (2009); Qian et al. (2009a); Qian et al. (2009b); Qian et al. (2009c)
2010	Tasgetien et al. (2010); Xu, Xiang e Wang (2010); Zheng e Yamashiro (2010)
2011	Mokhtari, Abadi e Cheraghalikhami (2011); Tasgetiren et al. (2011)
2012	Deng e Gu (2012); Dong e Wang (2012); Hu et al. (2012); Liu (2012); Lobato, Gedraite e Neiro (2012); Qian et al. (2012a); Qian et al. (2012b); Tasgetiren et al. (2012a); Tasgetiren et al. (2012b)
2013	Akrout et al.(2013a); Akrout et al. (2013b); Guimarães et al. (2013); Li e Yin (2013); Qian et al. (2013); Tasgetiren et al. (2013); Tonge e Kulkarni (2013)
2014	Amirian e Sahraeian (2014); Davendra et al. (2014); Liu; Yin e Gu (2014); Santuci, Baioletti e Milani (2014)
2015	Dong (2015); Tasgetiren et al. (2015); Tien et al. (2015)
2016	Santucci, Baioletti e Milani (2016); Shao e Pi (2016)

Fonte: o autor, 2017.

Uma síntese do conteúdo dos trabalhos, contemplando os principais aspectos dos problemas de programação investigados e dos algoritmos de ED desenvolvidos é a seguir apresentada.

A primeira aplicação da abordagem de ED para solução de PPPs em ambientes FSP, identificada nas bases de dados pesquisadas, foi apresentada por Tasgetien et al. (2004). No algoritmo de ED desenvolvido para minimização do *Makespan*, os autores incorporam a regra heurística SPVF (*Smallest Parameter Value First*) para determinar as permutações de tarefas. No algoritmo proposto procedimentos de busca local também são aplicados nos valores dos indivíduos contínuos e nas permutações de tarefas. A estratégia de ED utilizada é

ED/rand/1/bin e a inicialização da população ocorre de forma aleatória, conforme distribuição Uniforme. Os parâmetros *Np*, *CR* e *F* foram configurados conforme segue: *Np* = 2*n* (*n* = número de tarefas), *F* = 0,4 e *CR*= 0,5. O desempenho do algoritmo proposto foi comparado com os algoritmos AG (Algoritmo Genético) e PSO (*Particle Swarm Optimization*) e os resultados mostraram que os três algoritmos produzem resultados similares em termos de percentual relativo de aumento do *Makespan*. No entanto, em termos de tempo computacional, para problemas de pequeno porte, os tempos são similares, enquanto que, à medida que os problemas se tornam maiores, o PSO consome mais tempo que os algoritmos AG e ED.

Qian et al. (2006) apresentam um algoritmo de ED híbrido para o PPP em FSP, visando minimizar simultaneamente o *Makespan* e o Atraso Máximo. Para tornar o algoritmo de ED adequado ao problema, a regra LOV (*Largest-Order-Value*) baseada na representação de chaves aleatórias é desenvolvida para converter os valores contínuos dos indivíduos em permutações de tarefas. Para enriquecer o comportamento da busca e evitar convergência prematura, uma busca local baseada em VNS (*Variable Neighborhood Search*) é projetada e incorporada ao algoritmo. A estratégia de ED utilizada é *ED/rand-to-best/1/exp* e a inicialização da população ocorre de forma aleatória, conforme distribuição Gaussiana. Os parâmetros utilizados foram: *Np* = 2*n* (*n* = número de tarefas); *F* = 0,7 e 0,2; e *CR* = {0,1; 0,2; 0,4; 0,8; 1,0}. O algoritmo proposto é comparado com um AG denominado RWGA (*Random-Weight Genetic Algorithm*) proposto por Ishibuchi et al. (1998). Os resultados da experimentação computacional mostram que a qualidade do algoritmo proposto varia com a taxa de cruzamento (*CR*), sendo que a qualidade da solução diminui à medida que a *CR* aumenta.

Pan, Tasgetiren e Liang (2007) propõem um algoritmo denominado DDE (*Discrete Differential Evolution*) para a minimização do *Makespan*, em PPPs em ambientes FSP. Para melhorar a qualidade da solução, o algoritmo de ED é hibridizado com um procedimento de busca local RLS (*Referenced Local Search*) proposto por Rajendran e Ziegler (2004), passando a ser chamado de $DDE_{RLS.}$ A estratégia utilizada no algoritmo é *DE/rand/1/bin* e a inicialização da população ocorre de forma aleatória com distribuição uniforme. Os parâmetros utilizados são: *Np* = 20, *F* = 0,2 e *CR*= 0,8. Quando aplicado a problemas do *Benchmark* de Taillard e comparado com o algoritmo IG (*Iterated Greedy*) de Ruiz e Stutzle (2007) com

incorporação de busca local, o algoritmo DDE_{RLS} apresentou desempenho superior devido à inclusão da busca local.

Tasgetien et al. (2007) apresentam um algoritmo DDE (*Discrete Differential Evolution*) visando à minimização do Tempo de Fluxo Total em PPPs em ambientes FSP com restrições de *No-Wait*. Os autores também hibridizam o algoritmo DDE com uma busca local VND (*Variable Neighborhood Descendent*) proposta por Mladenovic e Hansen (1997) a fim de obter um melhor desempenho do algoritmo, o qual passa a ser denominado DDE_{VND}. A inicialização da população ocorre por meio das heurísticas NN (*Nearest Neighbor*) e NEH (*Nawaz, Enscore and Ham*), e a estratégia utilizada nos algoritmos é a *ED/rand/1/bin*. Os parâmetros utilizados foram: *Np: n* (*n* = número de tarefas); *F* = 0,8; *CR* = 0,8; *G* = 1000. Os algoritmos propostos são comparados com os algoritmos SA (*Simulated Annealing*), TS *Chins* (*Tabu Search* com *Cheapest Insertion*), TS *Pilot* (*Tabu Search* com *Pilot*) apresentados por Fink e Vob (2003) e com o algoritmo DPSO (*Discrete Particle Swarm Optimation*) apresentado por Pan et al. (2005). Os resultados computacionais mostram que o algoritmo DDE e DDE_{VND} superam os algoritmos SA, TS *Chins* e TS *Pilot* e apresentaram resultados superiores que o algoritmo DPSO. Os autores ressaltam que a inclusão da busca local VND no algoritmo DDE aumenta significativamente a qualidade da solução.

Pan, Tasgetiren e Liang (2008) apresentam extensões do algoritmo DDE (*Discrete Differential* Evoluiton) proposto por Pan, Tasgetiren e Liang (2007) e do algoritmo IG (*Iterated Greedy*) proposto por Ruiz e Stutzle (2007) para o caso de minimização do Tempo de Fluxo Total e do *Makespan* em PPPs em *FSP.* O procedimento de busca local RLS (*Referenced Local Search*) proposto por Rajendran e Ziegler (2004) é incorporado em ambos os algoritmos para melhorar a qualidade da solução. A inicialização de um indivíduo da população ocorre através da heurística NEH (*Nawaz, Enscore and Ham*) e os demais indivíduos são gerados aleatoriamente. A estratégia de ED utilizada nos algoritmos é *ED/rand/1/bin* e os parâmetros utilizados foram: *Np* = {10, 200, 2*n*}, com *n* = número de tarefas; *F* = 0,2; *CR*= 0,8. Os algoritmos apresentados foram comparados com os algoritmos disponíveis na literatura PSO_{SPV}, CPSO, GA_REEV, CPSO_PNEH, IG_{RIS}, DDE_{RLS}, PSO_{VNS}, H-CPSO, NEHT, GA_RMA, HGA_RMA, SA_OP, SPIRIT, GA_CHEN, GA_REEV, GA_MIT, ILS, GA_AA, M-MMAS, PACO, IG_RS, IG_RSLS, DDE sem

busca local e IG sem busca local e os resultados mostram que os dois algoritmos com busca local são melhores que os algoritmos da literatura.

Pan e Wang (2008) apresentam um algoritmo DDE (*Discrete Differential Evoluiton*) visando a minimização do *Makespan* em PPPs em ambientes FSP com restrições de *No-Idle.* Um algoritmo de busca local foi incorporado ao DDE para destacar o equilíbrio entre a exploração global e exploração local, passando a ser denominado DDE_{LS} (*Discrete Differential Evolution with Local Search*). A inicialização de um indivíduo da população ocorre por meio da heurística NEH (*Nawaz, Enscore and Ham*) e os demais indivíduos da população são gerados aleatoriamente. A estratégia de ED utilizada no algoritmo é *ED/rand/1/bin* e os parâmetros utilizados são: *Np:* 20; *F* = 0,8; e *CR* = 0,8. O desempenho dos algoritmos DDE e DDE_{LS} é comparado com os algoritmos IG, KK, DE/DE_{VNS} e PSO/ PSO_{VNS}, e os resultados mostram que os algoritmos DDE e DDE_{LS} são mais eficazes, eficientes e robustos do que os demais algoritmos. Em uma comparação entre o algoritmo DDE e DDE_{LS} foi possível notar que o algoritmo com a busca local obtém melhores resultados.

Qian et al. (2008) apresentam um algoritmo HDE (*Hybrid Differential Evolution*), que combina ED com busca local para o problema de programação em ambientes FSP, visando a minimização de diversos critérios de desempenho, como: *Makespan*, Máximo *Completion Time* Ponderado, Atraso Total, Atraso Total Ponderado, Máximo Adiantamento, Máximo Adiantamento Ponderado, Tempo Máximo de fluxo, Tempo de Fluxo Total Ponderado e Número de Tarefas Atrasadas. Primeiramente os autores propõem um algoritmo HDE, para minimização monocritério. Posteriormente, uma extensão do algoritmo HDE, denominada MHDE (*Multicriteria Hybrid Differential Evoluiton*), é proposta para a solução de problema multicritério. No HDE, o DE é aplicado para encontrar soluções promissoras sobre o espaço de soluções, e uma busca local simples para FSP é usada para explorar o espaço de solução e guiar a população para região onde o ótimo global ou o ótimo satisfatório está contido. A regra LOV (*Largest-Order-Value*) é utilizada para converter os valores contínuos dos indivíduos no DE em permutações de tarefas, de modo a tornar o DE adequado ao problema investigado. A inicialização da população ocorre de forma aleatória e a estratégia de ED adotada nos algoritmos é *ED/rand-to-best/1/exp*. Os parâmetros adotados foram: *Np* = 2*n*; *F* = 0,7; e *CR* = {0,1, 0,2, 0,4, 0,6, 0,8, 1,0}. O desempenho do HDE foi comparado com três

variantes do algoritmo de ED (HDE_NOL, HDE_ML e HDE_BL), e com os algoritmos ODE, PGA, NEH, OSA e HGA. O MHDE foi comparado com o algoritmo IMMOGLS2 (*Multi-Objective Genetic Local Search Algorithm*), desenvolvido por Yshibuchi, Yoshida e Murata (2003). Os resultados computacionais mostram a efetividade, a eficiência e a robustez dos algoritmos HDE e MHDE propostos.

Pan, Wang e Qian (2009) apresentam um algoritmo DDE (*Discrete Differential Evoluiton*) para o problema de minimização do *Makespan* e do Atraso Máximo em ambientes FSP com restrições de *No-Wait*. No algoritmo apresentado, novos operadores de mutação e cruzamento são incorporados, e um operador de seleção um-a-um é projetado levando em consideração o estado de dominância de um indivíduo alvo. A fim de obter um bom conjunto de soluções, o algoritmo DDE atualiza dinamicamente o conjunto de soluções não-dominadas durante o processo de busca. Vários métodos de aceleração são concebidos para avaliar uma permutação de tarefas e toda a sua vizinhança de inserção, bem como para determinar o estado de dominância de uma solução em relação às soluções do conjunto arquivo. A inicialização da população ocorre por meio da heurística PWQ baseada nas heurísticas NEH e EDD e a estratégia de ED adotada no algoritmo é *ED/rand/1/bin.* Os parâmetros adotados foram: $Np = 5n$; $F = 0{,}2$; e $CR = 0{,}2$. O algoritmo DDE proposto foi comparado com o algoritmo HDE de Qian et al. (2006) e com o algoritmo IMMOGLS2 (*Multi-Objective Genetic Local Search Algorithm*) desenvolvido por Yshibuchi, Yoshida e Murata (2003). Os resultados da experimentação computacional e das comparações mostraram que o DDE proposto é superior em termos de qualidade de busca, nível de diversidade, robustez e efetividade.

Qian et al. (2009a) apresentam um algoritmo HDE (*Hybrid Differential Evoluiton*), que combina Evolução Diferencial com busca local para o problema de minimização do *Makespan* em ambientes FSP com restrições de *No-Wait*. No algoritmo proposto, a regra LOV (*Largest-Order-Value*) é utilizada para transformar os valores reais dos indivíduos do ED em permutações de tarefas, um mecanismo de evolução paralela é aplicado para executar eficientemente a exploração e uma busca local é desenvolvida. Um método de aceleração da avaliação e um método baseado no exame rápido da vizinhança de inserção são desenvolvidos com base nas propriedades do problema de programação em ambientes FSP com restrições de *No-Wait*. A inicialização da população ocorre de forma aleatória e a estratégia de

ED adotada pelos autores é a *ED/rand-to-best/1/exp*. Os parâmetros utilizados são: *Np* = 50; *F* = 0,7; e *CR* = {0,1; 0,8}. Os testes computacionais para análise da regra LOV contemplam comparações entre o HDE com a regra LOV sem o método de aceleração com busca local, o HDE sem a regra LOV sem o método de aceleração com busca local e o método RAJ, proposto por Rajendran (1994). Para ambas as taxas de cruzamento, a heurística HDE com a regra LOV sem o método de aceleração com busca local apresentou o melhor desempenho. Para encontrar uma busca local para o HDE, três variantes do HDE (HDE_KS5, HDE_*Insert+Insert*, HDE_VNS) com busca local foram testadas, e o HDE_KS5 obteve os melhores desempenhos. Para comprovar a eficiência e a efetividade do HDE, o mesmo foi comparado com as instancias de Reeves (1995) configuradas com o número máximo de gerações do OSA (Algoritmo *Simulated Annealing*) e com o algoritmo HPSO (*Hybrid Particle Swarm Optimization*) proposto por Liu et al. (2007), e os resultados mostram que o HDE é superior ao OSA e ao HPSO para quase todas as instancias testadas.

Qian et al. (2009b) desenvolveram um algoritmo baseado em ED para resolver o problema de programação multi-objetivo em ambientes FSP com *buffers* limitados. A regra LOV (*Largest-Order-Value*) foi adotada para converter indivíduos contínuos em permutação de tarefas e o conceito de dominância de Pareto foi usado para lidar com a atualização das soluções multi-objetivo. Uma busca local dependente do problema foi projetada com base nas características do problema de programação em *Flow Shop* Permutacional com *buffers* limitados e aplicada para enfatizar a exploração. No algoritmo de ED desenvolvido pelos autores a inicialização da população ocorre de forma aleatória, a estratégia de ED adotada é a *ED/rand-to-best/1/exp* e os parâmetros utilizados são: *Np* = 50; *F* = 0,7; *CR* = 0,1; e bG_{max} = 300. Simulações de instancias dos *Benchmarks* de Carlier, Reeves e Taillard e comparações com o algoritmo IMMOGLS2 (*Multi-Objective Genetic Local Search Algorithm*) proposto por Gourgand, Grangeon e Norre (2005) demonstraram a eficiência e a efetividade do algoritmo.

Qian et al. (2009c) propõem um algoritmo denominado HDE (*Hybrid Differential Evoluiton*) para resolver o problema de programação em ambientes FSP com *buffers* limitados. No algoritmo HDE proposto, a regra LOV (*Largest-Order-Value*) é utilizada para transformar os valores reais dos indivíduos do DE em permutações de tarefas e uma busca local dependente do problema e baseada no

conceito de blocos (NOWICKI, 1999) é desenvolvida e aplicada para enfatizar a exploração. No algoritmo de HDE desenvolvido pelos autores a inicialização da população ocorre de forma aleatória com distribuição uniforme, a estratégia de ED adotada é a *ED/rand-to-best/1/exp* e os parâmetros utilizados são: *Np* = 30; *F* = 0,7; e *CR* = 0,1. Para testar o desempenho do algoritmo HDE proposto para *Flow Shop* Permutacional com *buffers* limitados, três variantes do HDE foram comparadas: HDE_NOL (HDE sem busca local); HDE_ML (HDE com 1/5 dos indivíduos selecionados aleatoriamente em cada geral para execução da busca local); e HDE_BL (HDE com busca local baseada no conceito de blocos). Em adição, o HDE proposto também teve seu desempenho comparado com o algoritmo OSA (OSMAN e POTTS, 1989). Os resultados das experimentações computacionais demonstraram a eficiência e a efetividade do algoritmo.

Tasgetien et al. (2010) apresentam um método que hibridiza o algoritmo DABC (*Discrete Artificial Bee Colony*), o algoritmo IG (*Iterated Greedy*), o algoritmo ILS (*Iterated Local Search*) e o algoritmo VNS (*Variable Neighborhood Search*) para o problema de minimização do Tempo de Fluxo Total em ambientes FSP. Os autores também hibridizam um algoritmo HDDE (*Hybrid Discrete Differential Evoluiton)* com um algoritmo IG e um algoritmo VNS. Na estratégia de ED adotada nos algoritmos o mutante individual é obtido por meio da perturbação de cada indivíduo da população alvo. A inicialização de um indivíduo da população ocorre através da heurística NEH e os demais indivíduos da população são gerados aleatoriamente. Os parâmetros adotados nos algoritmos são: *Np* = 10; *F* = 0,2*;* e *CR* = 0,9. O desempenho dos algoritmos DABC e HDDE propostos pelos autores são comparados com os algoritmos HGLS, VNS e EDA. Os resultados computacionais mostram que o desempenho do DABC e do HDDE é altamente competitivo com os algoritmos EDA e HGLS, tanto em termos de qualidade da solução, quanto em tempo computacional.

Xu, Xiang e Wang (2010) propõem um algoritmo de ED Autoadaptativo para o problema de minimização do *Makespan* em ambientes FSP. No algoritmo proposto, baseado na Teoria do Caos, os parâmetros *F* e *CR* são ajustados adaptativamente para cada indivíduo de acordo com seu valor de aptidão e a regra LOV (*Largest-Order-Value*) é adotada para converter indivíduos contínuos em permutação de tarefas. A inicialização da população ocorre de forma aleatória e a estratégia de ED adotada no algoritmo é *DE/rand/1/bin*. Os parâmetros adotados são: *Np* = 5*n*; *F* =

0,8; e *CR* = 0,5. O desempenho do algoritmo proposto pelos autores é comparado com os algoritmos ED e HDE_NOL (*Hybrid Differential Evolution without Local Search*). Os resultados computacionais mostram a eficiência e a eficácia do algoritmo proposto.

Zheng e Yamashiro (2010) para resolver o PPP em ambientes FSP visando minimizar o *Makespan*, Tempo de Fluxo e Atraso Máximo dos Postos de Trabalho apresentam um algoritmo QDEA (*Quantum Differential Evolutionary Algorithm*) baseado no algoritmo QEA (*Quantum-Inspired Evolutionary*). Os autores hibridizam o QDEA com o algoritmo VNS (*Variable Neighborhood Search*) para aumentar o desempenho da busca local. No algoritmo QDEA, os cromossomos quânticos são codificados e decodificados usando o ângulo rotativo quântico e uma simples estratégia denominada "regra do valor do maior ângulo rotativo" para determinar a sequência de tarefas. A estratégia de ED adotada no algoritmo é *ED/rand/2/exp* e os parâmetros são configurados conforme segue: *Np* = 50; *F* = 0,9; e *CR* = 0,1. O desempenho do algoritmo proposto pelos autores é comparado com os algoritmos PSO, DPSO, HQEA, HGA, HDE, BEST(LR), M-MMAS, ACO, PACO, LS, PSO-LS e três estratégias de busca local (VNS1, VNS2 e VNS3). Os resultados computacionais mostram a eficácia do método proposto.

Mokhtari, Abadi e Cheraghalikhami (2011) investigam o problema de tempos de processamento dependentes de recursos (RDPT) em FSP em que o tempo de processamento de um trabalho depende da quantidade de recursos adicionais atribuídos a esse trabalho. Os autores apresentam um algoritmo HDDE (*Hybrid Discrete Differential Evolution*) para minimizar o *Makespan* e a Quantidade Necessária de Recursos. No algoritmo de desenvolvido, os autores incorporam um novo operador híbrido cíclico para melhorar a qualidade da solução e diminuir o período de convergência. Uma busca local VNS (*Variable Neighborhood Search*) também é aplicada no HDDE para melhorar o seu desempenho. A inicialização da população ocorre através das heurísticas SPT, LPT, SOT e LOT e aleatoriamente. Para identificar e definir os parâmetros significativos do algoritmo de ED apresentado que utilizada a estratégia *ED/rand/1/bin* é realizado um procedimento estatístico com base na técnica de regressão. Os parâmetros do algoritmo foram configurados conforme segue: *Np* = 40, 100 e 300; *F* = 0,9; *e CR* = 0,05. O desempenho do algoritmo com busca local proposto pelos autores é comparado com os algoritmos ODE, NEH, PGA, CDS, Palmer e HDDE sem busca local. Os

resultados computacionais mostram a eficácia do HDDE com busca local em relação aos outros métodos.

Tasgetiren et al. (2011a) investigam o problema de minimização do *Tardiness* em ambientes FSP com restrições de *No-idle*. Um algoritmo de ED com busca variável de parâmetros (vpsDE) é desenvolvido e comparado com o algoritmo RKGA, um algoritmo genético de chave aleatória. No algoritmo vpsDE, a regra LPV (*Largest Parameter Value*) é adotada para converter indivíduos contínuos em permutação de tarefas, a inicialização da população ocorre de forma aleatória e a estratégia de ED adotada no algoritmo é *DE/rand/1/bin*. Os parâmetros do algoritmo foram fixados em: Np = 30; DE_1 [F = 0,9; CR = 0,9]; DE_2 [F = 0,1; CR = 0,9]; DE_3 [F = 0,9; CR = 0,1]; DE_4 [F = 0,1; CR = 0,1]; e T_{max} = 10n milissegundos. Simulações computacionais com instancias do *Benchmark* de Taillard mostraram que o algoritmo de ED com busca variável de parâmetros (vpsDE) é muito promissor para resolver o problema *Flow Shop* Permutacional com restrições de *No-idle* e gera resultados estatisticamente melhores que o algoritmo RKGA.

Tasgetiren et al. (2011b) apresentam um algoritmo DABC (*Discrete Artificial Bee Colony*) para o problema de minimização do Tempo Total de Fluxo em ambientes FSP. Os autores também apresentam uma extensão do algoritmo HDDE (*Hybrid Discrete Differential Evolution*) proposto por Tasgetiren et al. (2011a). Os autores hibridizam esses algoritmos com uma variante do algoritmo IG (*Iterated Greedy*), denominada IG_RS. Nos algoritmos apresentados, a inicialização de um indivíduo da população ocorre através da heurística NEH e o restante dos indivíduos da população são gerados aleatoriamente com distribuição uniforme. A estratégia de ED adotada é a *ED/rand/1/bin* e os parâmetros dos algoritmos são configurados conforme segue: Np = 10; F = 0,5; e CR = 0,9. O desempenho dos algoritmos é testado sobre os problemas do *Benchmark* de Taillard, e também é comparado com os algoritmos EDA,VNS, TSGLS e HGLS. As comparações mostram que os algoritmos HDDE e DABC apresentam melhores resultados para a maioria dos problemas testados.

Deng e Gu (2012) apresentam um novo método de aceleração baseado em representação de redes, que reduz a complexidade das buscas e o incorporam a uma heurística HDDE (*Hybrid Discrete Differential Evolution*) para resolver problemas de programação em ambientes FSP com restrições de *No-Idle*, visando minimizar o *Makespan*. Além disso, uma busca local de inserção de Vizinhança é

modificada no algoritmo de ED para equilibrar a exploração global e a exploração local. O novo método de aceleração proposto é usado no NEH (Nawaz, Enscore e Ham) para geração de soluções iniciais e na busca de inserção de vizinhança. Na inicialização da população, um, indivíduo é gerado por meio da Heurística NEH, um indivíduo é gerado por uma variante da *Insertion Improvement* e os demais indivíduos são gerados aletoriamente. A estratégia de ED utilizada no HDDE é *ED/rand/1/bin* e os parâmetros adotados são: *NP* = 20; *F* = 0,8; e *CR* = 0,8. Na avaliação computacional, o algoritmo HDDE proposto é comparado com as heurística NEHD (NEH reimplementada pelos autores com o método de aceleração) e NEH_{nsu} (NEH sem o método de aceleração) e com as metaheurísticas IG_{LS}, HDPSO e DDE_{LS}, ambas reimplementadas pelos autores com os parâmetros originais, exceto pelo fato de que o método de aceleração proposto foi incorporado às heurísticas. Os resultados computacionais mostraram que a HDDE é superior aos outros algoritmos para o problema de programação em FSP *No-Idle* com critério *Makespan.*

Hu et al. (2012) apresentam um algoritmo de ED, denominado DE_NTJ, para minimizar o número de tarefas em atraso em ambientes de programação FSP *No-Wait*, com tempos de *setup* dependentes da sequência e diferentes datas de liberação. Para equilibrar as habilidades de prospecção e exploração, o algoritmo de ED baseado em busca global é utilizado para obter as regiões ou soluções promissoras sobre o espaço de solução. Propriedades do problema e uma busca local são desenvolvidas para explorar as vizinhanças destas regiões. No algoritmo proposto a regra LOV (*Largest-Order-Value*) é utilizada para converter os indivíduos contínuos em indivíduos discretos e um método de aceleração de avaliação das soluções também é utilizado. No algoritmo proposto, a estratégia adotada é a *ED/rand-to-best/1/exp* e a inicialização da população ocorre de modo aleatório. Os parâmetros para foram configurados conforme segue: *Np* = 30, *F* = 0,7 e *CR* = 0,1. Na experimentação computacional o algoritmo DE_NTJ foi comparado com os algoritmos IG e HDE. O algoritmo também foi comparado com duas variantes do DE_NTJ, o $DE\_NTJ\_V_1$ e o $DE\_NTJ\_V_2$, que se diferenciam no modo de exploração da busca local incorporada. Os resultados mostraram a superioridade do DE_NTJ em termos de qualidade de busca, eficiência e robustez.

Liu (2012) apresenta um algoritmo de ED para o problema de minimização do Tempo de Fluxo Total em ambientes FSP. Para aumentar a capacidade de

exploração de ED, os autores incorporam o SA (*Simulated Annealing*) e o VNS (*Variable Neighborhood Search*) no algoritmo. A regra SPV (*Smallest Position Value*) é adotada para converter uma solução contínua em uma solução discreta. No algoritmo apresentado, a estratégia de ED utilizada é a DE/*rand/1/bin* e a inicialização da população ocorre através da heurística NEH e aleatoriamente. O parâmetro *CR* é aaleatório baseado em distribuição Gaussiana com média 0,5 e desvio padrão 0.1 e então truncado em [0,1], o parâmetro *F* é igual a 0,5 e o tamanho da população (*Np*) é igual a 100. O algoritmo proposto foi comparado com os algoritmos PSO_{VNS} e H-CPSO. Os resultados mostram que o algoritmo desenvolvido pelos autores supera os demais algoritmos.

Lobato, Gedraite e Neiro (2012) testam a eficiência do algoritmo de ED para minimização do *Makespan* em um ambiente FSP, variando os valores dos parâmetros do algoritmo e analisando os resultados correspondentes com uma estratégia clássica implementada no *software* GAMS *General Algebraic Modelling System* (GAMS). No algoritmo proposto, a estratégia adotada é a *ED/rand/1/bin* e a inicialização da população ocorre de modo aleatório. Os parâmetros para foram configurados conforme segue: *Np* = 500; *F* = {0,1, 0,5, 0,8, 1,0, 2,0}; e *CR* = {0,1, 0,3, 0,5, 0,8, 0,9}. Os resultados computacionais mostram a superioridade do algoritmo proposto em relação ao da abordagem clássica.

Qian et al. (2012a) para o problema de minimização do *Total Completion Time* em ambientes FSP *No-Wait*, com tempos de *setups* dependentes da sequencia e diferentes datas de liberação, apresentam um algoritmo de ED, denominado DE_TSE, que incorpora dois métodos de aceleração e uma busca local rápida. No algoritmo proposto a regra LOV (*Largest-Order-Value*) é adotada para converter os indivíduos contínuos em permutações de tarefas. A estratégia de ED adotada no algoritmo é *ED/rand-to-best/l/exp* e os parâmetros adotados são: *Np* = 30; F = 0,7; e CR = 0,1. Os resultados do DE_TSE foram comparados com os resultados fornecidos pelos algoritmos HDE, SAFM e IG. Os resultados mostraram que sob a mesma quantidade de tempo computacional, o DE_TSM pode pesquisar mais regiões /soluções, o que é útil para a obtenção de um melhor desempenho.

Qian et al. (2012b) apresentam um algoritmo híbrido baseado em Evolução Diferencial para o problema de minimização do Tempo Total de Conclusão em ambientes FSP com restrições de *No-Wait*, tempos de *setup* dependentes da sequencia e diferentes datas de liberação. No algoritmo proposto, denominado HDE

(*Hybrid Differential Evolution*), a regra LOV (*Largest-Order-Value*) é utilizada para transformar os valores contínuos dos indivíduos do ED em permutações de tarefas. Neste estudo um método de aceleração de avaliação é desenvolvido de acordo com as propriedades do *No-Wait Flow Shop* com tempos de *setup* dependentes da sequencia e diferentes datas de liberação. No HDE proposto, o ED é usado para encontrar as soluções ou regiões promissoras sobre o espaço da busca e, em seguida, uma busca local e um método de aceleração de avaliação são incorporados ao algoritmo para explorar o espaço de solução dessas regiões. No HDE a inicialização da população ocorre de forma aleatória, a estratégia de ED adotada é a *ED/rand-to-best/1/exp* e os parâmetros utilizados foram: *Np* = 30; *F* = 0,7; e *CR* = 0,1. Comparações do HDE com os algoritmos SAFM (ISHIBUCHI, MISAKI e TANAKA, 1995), HDE_FSSP (QIAN et al., 2008), HDE_FSSP_SP (HDE_FSSP com método de aceleração de avaliação) e IG (RUIZ e STÜTZLE, 2008) mostraram que o HDE é uma abordagem eficaz com excelente qualidade e robustez para problemas de programação em ambientes FSP com restrições de *No-Wait*, tempos de *setup* dependentes da sequencia e diferentes datas de liberação.

Tasgetiren et al. (2012a) apresentam um algoritmo vIGP_DE em que os parâmetros do IG (tamanho da destruição e arrefecimento) são otimizados pelo algoritmo de ED. Uma única solução multi-cromossomo, de modo que o primeiro cromossomo representa o tamanho da destruição e arrefecimento, enquanto o segundo representa uma permutação de tarefas designada aleatoriamente a cada individuo da população é apresentada pelos autores. O algoritmo proposto é aplicado ao problema de programação em *Flow Shop No-Idle* com critério de Tempo Total de Fluxo (*Total Flow Time*). A estratégia de ED adotada no algoritmo é *ED/rand/1/bin* e os parâmetros adotados são: *Np* = 30; *F* =0,9 e *CR* =0,9. O desempenho do algoritmo foi testado no *Benchmark* de Taillard e os resultados foram comparados com resultados os fornecidos pelo algoritmo vIG_FR. O vIGP_DE forneceu as melhores soluções para 63 dos 120 problemas testados e forneceu os mesmos resultados que o vIG_FR para 15 dos 120 problemas testados.

Tasgetiren et al. (2012b) investigam o problema de minimização do *Makespan* em ambientes FSP com restrições de *No-Idle* e propõem um algoritmo, denominado vIG_DE, que combina Variável Iterada Gulosa com ED. No algoritmo proposto os parâmetros (tamanho da destruição e probabilidade de aplicar ou não o IG a um indivíduo) são otimizados por meio do algoritmo de Evolução Diferencial. Os autores

apresentam uma representação de uma única solução multi-cromossomo, de modo que o primeiro cromossomo representa o tamanho da destruição e probabilidade, enquanto o segundo representa uma permutação de tarefas designada aleatoriamente a cada individuo da população. A estratégia de ED adotada no algoritmo é *ED/rand/1/bin* e os parâmetros adotados foram: *Np* = 30; *F* =0,9 e *CR* =0,9. O desempenho do algoritmo foi testado no *Benchmark* de Ruiz e os resultados foram comparados com os resultados fornecidos pelos algoritmos IG_LS e HDDE. O vIG_DE forneceu as melhores soluções para183 dos 250 problemas testados e forneceu os mesmos que o IG_LS resultados para 49 dos 250 problemas testados. Quando comparado com os resultados do HDDE, o vIG_DE melhorou 114 das 150 soluções, conhecidas como melhores soluções, com metade do tempo computacional alocado.

Akrout et al. (2013a) e Akrout et al. (2013b) examinam o PPP em ambientes FSP com restrições de *No-Wait*, tendo o *Makespan* como critério de desempenho. Os autores propõem uma heurística híbrida, denominada NEW-GRASP-DE (*New-Greedy Randomized Adaptive Search Procedure-Differential Evolution*) que combina o algoritmo GRASP com o algoritmo de ED. No algoritmo proposto todos os parâmetros do GRASP são ajustados pelo algoritmo de ED, e no GRASP um algoritmo ILS (*Iterated Local Search*) é usado como um procedimento de melhoria. Os parâmetros adotados para o ED foram *Np* = 50; *F* = 1,5 e *CR* =0,85 e a estratégia de ED adotada foi a *ED/rand/1/bin*. O desempenho do algoritmo foi testado nos *Benchmarks* de Taillard, Reeves e Heller e os resultados foram comparados com os resultados fornecidos pelos algoritmos RAJ, VNS_{SF}, $GASA_{SF}$, DS_{GP}, $DS+M_{GP}$, TS_{GP}, $TS+M_{GP}$, $TS+MP_{GP}$, HGA_{TL}, $DPSO_{PTL}$, $DPSOVND_{PTL}$ e $GA-VNS_{JES}$. De acordo com os resultados da experimentação computacional o algoritmo NEW-GRASP-DE mostrou-se altamente competitivo. Os estudos de Akrout et al. (2013a) e Akrout et al. (2013b) diferenciam-se no critério de parada utilizado no algoritmo NEW-GRASP-DE. O tempo máximo computacional e o número máximo de iterações, são definidos como critério de parada do algoritmo em Akrout et al. (2013a) e (2013b), respectivamente.

Guimarães et al. (2013) apresentam um algoritmo DDE (*Discrete Differential Evolution*) para problemas de otimização combinatória. As ideias do algoritmo proposto são aplicadas a um problema de programação de *n*-tarefas e *m*-máquinas em ambientes FSP. No algoritmo proposto uma busca local RLS (*Referenced Local*

Search) é aplicada ao melhor indivíduo de cada geração. As estratégia de *ED/rand/1/bin* e *ED/best/1/bin* foram utilizadas nos testes computacionais do algoritmo proposto. No algoritmo proposto, o tamanho da população foi fixado em *Np* = 10 e a cada vez que um indivíduo passa pela fase de mutação e de cruzamento, é selecionada aleatoriamente uma das combinações *F* e *CR*: [*F* = 1, *CR* = 0,1], [*F* = 1, *CR* = 0,9] e [*F* = 0,8, *CR* = 0,2]. A estratégia *ED/rand/1/bin* foi executada com probabilidade 0.8 e a estratégia *ED/best/1/bin* com probabilidade 0,2. O operador de cruzamento usando foi o OX, proposto por Davis (1985). O algoritmo foi aplicado a 120 instancias do *Benchmark* de Taillard (1993) e comparado com os algoritmos IG_RS, IG_RS$_{LS}$, NEHT, GA_RMA, AS_OP, SPIRIT, HGA_RMA, GA_CHEN, GA_REEV, GA_MIT, ILS, GA_AA, M-MMAS, PACO e DDERLS. Os resultados mostram que a abordagem de ED constitui uma aplicação promissora para problemas de otimização combinatória.

Li e Yin (2013) propõem um novo algoritmo de ED baseado em algoritmo Memético, denominado ODDE, para resolver o PPP em ambientes FSP com função objetivo monocritério, sendo definidos para o estudo os critérios de *Makespan* e o Máximo *Lateness*. Os autores introduzem uma nova regra para converter a posição contínua no DE em permutação discreta de tarefas, a regra LRV (*Largest-Ranked-Value*), baseada em chaves aleatórias (*Randon Key*). A heurística NEH é combinada com inicialização aleatória para gerar a população inicial. Para melhorar a propriedade de otimização global do ED, um enfoque baseado no grau de diversidade da população é proposto para afinar a taxa de cruzamento e para melhorar a taxa de convergência do ED, um procedimento de busca baseado em oposição é aplicado, com o propósito de melhorar a solução global ótima. Um esquema de busca local também é incorporado ao algoritmo. A estratégia adotada foi a *ED/target-to-best/1* e os parâmetros para foram configurados conforme segue: *Np* = {60, 2*n*}, *F* = 0,7 e *CR* ajustado adaptativamente. O algoritmo ODDE proposto foi comparado com os algoritmos DE, HDE, OSA, PSOMA, PSOVNS, SGA, SGA+NEH, HGA, HQEA, HQDEA, ATPPSO, L-CDPSO e LWK_SA1, e os resultados da experimentação computacional mostraram que o algoritmo proposto pode geralmente produzir melhores resultados que os outros algoritmos. Para os problemas com o objetivo de minimizar o *Makespan*, o algoritmo ODDE obteve 24 novos limitantes superiores das 40 instâncias testadas, e para os problemas com o

objetivo de minimizar o Máximo *Lateness*, o algoritmo ODDE, obteve 137 novos limitantes superiores das 160 instâncias testadas.

Qian et al. (2013) investigam o problema de minimização do *Makespan* e do em ambientes FSP com fluxo reentrante de tarefas. Um algoritmo de ED, denominado DE_TST, que combina uma busca local, com uma estratégia de aceleração e uma estratégia de mudança de vizinhança é proposto para resolver o problema. No algoritmo proposto a regra LOV (*Largest-Order-Value*) é utilizada para converter os indivíduos contínuos do ED em indivíduos discretos (permutações de tarefas). No algoritmo de ED proposto, a inicialização da população ocorre de modo aleatório, a estratégia de ED adotada foi a *ED/rand-to-best/1/exp* e os parâmetros foram configurados conforme segue: *Np* = 30, *F* = 0,7 e *CR* = 0,1. O algoritmo DE_TST proposto foi comparado com os algoritmos HGA, MRPFSSP e DE_TST_ V, uma variante do algoritmo proposto que apresenta uma alteração na fase de exploração da busca local incorporado ao DE_TST. Os resultados mostram que o DE_TST é um algoritmo efetivo para o problema em questão.

Tasgetiren et al. (2013) apresentam um algoritmo vIG_DE, que mescla o algoritmo VIG (*Variable Iterated Greedy*) com o algoritmo de ED para resolver o PPP em ambientes FSP com restrições de *No-Idle*, com função objetivo monocritério, sendo considerados no estudo os critérios de *Makespan* e o Máximo *Lateness*. No algoritmo proposto o ED é empregado para determinar dois importantes parâmetros do IG, o tamanho da destruição e a probabilidade de aplicação do IG em um indivíduo. A estratégia de ED adotada no algoritmo é *ED/rand/1/bin* e os parâmetros adotados são: *Np* = 30; *F* =0,9 e *CR* =0,9. O algoritmo proposto foi comparado com os algoritmos HDDE, DDE_LS, HDPSO, IG_LSIG_RIS, VIG_FL e os resultados demonstraram a superioridade do algoritmo proposto.

Tonge e Kulkarni (2013) apresentam três diferentes abordagens de ED para minimização do *Makespan* em ambientes FSP. A primeira abordagem, denominada DEPCO, consiste em um algoritmo de ED com operador de cruzamento baseado em posição. A segunda abordagem, denominada DE_NEH-ILS-ESP, consiste em algoritmo de ED, que combina os algoritmos NEH (*Nawaz, Enscore and Ham*) e ILS (*Iterated Local Search*). A terceira abordagem consiste em um algoritmo de ED melhorado, onde a população após ser avaliada é classificada em ordem crescente, dividida em subpopulações e o ED é aplicado as subpopulações até a obtenção do mínimo. Nas abordagens propostas, a estratégia adotada foi a *ED/rand/1/bin* e a

inicialização da população foi realizada de modo aleatório e os parâmetros para foram configurados conforme segue: *Np* = {400, 700, 2500, 7000, 9000}, *F* = {0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9} e *CR* = {0,1, 0,3, 0,6, 0,7, 0,8}. Os valores dos parâmetros variam conforme as abordagens testadas. Além de comparações entre as três abordagens propostas, foram efetuadas comparações com os algoritmos GA e QIDE. Os resultados mostraram que o DEPCO foi capaz de obter a solução ótima para os 8 problemas do *benchamark* de Carlier testados na experimentação computacional.

Amirian e Sahraeian (2014) examinam o PPP em ambientes FSP com um efeito de aprendizagem modificado em um ambiente bicritério, em que os critérios são o *Makespan* e o Tempo Total de Conclusão (*Total Completion Time*). Inicialmente foi proposta para o problema de *Flow Shop* uma versão truncada do efeito de aprendizagem de Dejong. Devido à elevada complexidade do modelo, foi proposto um algoritmo de ED híbrido, denominado MDES, para resolver o problema. O MDES combina a diversidade dos métodos de triagem e seleção de soluções não dominadas do algoritmo NSGA-II, com o ED clássico e introduz uma população avançada para armazenar as soluções mais desejáveis de cada iteração, melhorando deste modo a diversidade. Uma busca local baseada em inserção é incorporada na fase de inicialização e no ciclo principal do algoritmo para melhorar a exploração. No algoritmo proposto, MDES, a estratégia adotada foi a *ED/best/1/bin*, no entanto, na experimentação computacional a estratégia *ED/rand-to-best/1/exp* também foi testada. A inicialização da população foi realizada através de uma busca local baseada em inserção e os parâmetros para foram configurados conforme segue: *Np* = 100, *F* = {0,6023, 0,80233} e *CR* = 0,2. Os resultados do MDES foram comparados com os resultados fornecidos pelos algoritmos PASA, MPFA, MOSA e PGA-ALS. Os resultados e as métricas de desempenho computacional mostraram a eficiência do método proposto.

Davendra et al. (2014) analisam os atributos da dinâmica populacional de algoritmo de ED usando ferramentas de Análise de Redes Complexas. A população é visualizada como uma rede complexa em evolução, a qual exibe características não triviais. Uma rede complexa contêm recursos, que são únicos para o problema, como o grau de distribuição, agrupamento de dados (*clustering*), estruturas comunitárias, entre outros, que são marcadores importantes para a AEs. De acordo com os autores pode ser postulado que uma população sob jurisdição de um AE

apresenta tal comportamento de rede complexa. Isto é reforçado pela aplicação de um AE, uma vez que a rede é projetada para interligar a população, a fim de gerar novos indivíduos. Para demonstrar o uso de redes complexas os autores utilizaram o algoritmo EDE (Davendra e Onwubolu, 2009) e consideram o problema de minimização do *Makespan* em ambientes FSP com restrições de *No-Wait*. Neste estudo a abordagem de adjacência ponderada em grafo é utilizada com o intuito de mostrar o relacionamento entre diferentes indivíduos na população. A experimentação computacional foi conduzida no *software* Mathematica 9, utilizando o conjunto de ferramentas de análise de rede social. O tamanho da população foi definido em 100 indivíduos com 100 gerações. A partir da análise da população como um grafo de adjacência ponderada, ficou claro que a população do EDE, cuja estratégia de ED adotada foi a *ED/best/1/exp,* apresenta comportamento de redes complexas durante as avaliações.

Liu, Yin e Gu (2014) propõem um algoritmo de ED híbrido, denominado L_HDE, para resolver o problema de minimização do *Makespan* em ambientes FSP. O L-HDE combina o ED com a busca local IIS (*Individual Improving Scheme*) e a busca local GB (*Greedy-Based*). A regra LRV (*Largest-Ranked-Value*), baseada em chaves aleatórias (*Randon Key*) é introduzida para converter a posição contínua no ED em permutação de tarefas. A heurística NEH é combinada com inicialização aleatória com distribuição uniforme para gerar a população inicial. A estratégia adotada foi a *ED/rand/1/bin* e os parâmetros para foram configurados conforme segue: $Np = \{60, 2n\}$, $F = 2$ e $CR = 0{,}4$. O algoritmo L-HDE proposto foi comparado com os algoritmos DE, HDE, ATPPSO-R, ATPPSO, HPSO, NPSO, PSOMA, PSOVNS, SGA, SGA+NEH e HGA e os resultados experimentais mostraram que o algoritmo proposto pode geralmente produzir melhores resultados que os outros algoritmos.

Santuci, Baioletti e Milani (2014) propõem um algoritmo DDE (*Discrete Differential Evolution*) para o problema de minimização do tempo total de fluxo (*Total Flow Time*) em FSP, que atua diretamente sobre o espaço de permutações. O núcleo do algoritmo proposto, denominado DEP, é o operador de mutação diferencial com base na distância definida por meio de um novo algoritmo *bubble sort* randomizado. O esquema de mutação permite que o algoritmo de ED navegue diretamente nas permutações do espaço de busca. No DEP o cruzamento é realizado de acordo com a versão II do cruzamento de dois pontos (TPII) proposto

por Murata, Ishibuchi e Tanaka (1996) e usado no AGA de Xu, Xu e Gu (2011). No DEP, a estratégia adotada é a *ED/rand/1/bin* e a inicialização de um indivíduo da população ocorre através da heurística construtiva *LR*(*n/m*) e o restante dos indivíduos da população são gerados aleatoriamente. O parâmetro *F* é ajustado adaptativamente de acordo com o procedimento do algoritmo jDE e o tamanho da população (*Np*) é igual a 100. O desempenho do algoritmo DEP foi comparado com o desempenho dos algoritmos AGA, VNS_4, GM_EDA e HGM_EDA. Os resultados mostraram que o algoritmo de DEP proposto apresenta melhores resultados para a maioria dos problemas testados.

Dong (2015) apresentam um algoritmo de ED, denominado HPDETL, para o problema de minimização do *Makespan* em ambientes FSP com restrições de *No-Wait*. O algoritmo proposto HPDETL emprega um novo algoritmo de ED discreto com lista tabu (PDETL) para gerar novos indivíduos. Uma busca local baseada na heurística complexa rápida (PCH) é embutida no PDETL para melhorar o desempenho da busca. A estratégia de ED adotada no algoritmo é *ED/rand/1/bin* e os parâmetros adotados são: *Np* = {30, *n*}; *F* = 0,5 *e CR* = 0,5. Os resultados produzidos pelo algoritmo proposto foram comparados com os limitantes superiores fornecidos pelo algoritmo RAJ e com os resultados produzidos pelos algoritmos HDE e $DPSO_{VNS}$, demonstrando que o novo algoritmo é eficiente e eficaz com excelente qualidade e robustez para o problema em questão.

Tasgetiren et al. (2015) apresentam um algoritmo de Busca Local Populada através do algoritmo de ED para resolver o PPP em ambientes FSP com restrições de bloqueio e minimização do *Makespan*. No algoritmo proposto, denominado DE_PLS, o algoritmo de ED é responsável pelo fornecimento dos parâmetros dos algoritmos IG e ILS. Uma nova representação de solução multi-cromossomo para a busca local e um algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*) também são propostos neste estudo. No algoritmo proposto, a estratégia adotada é a *ED/best/1/bin* e a inicialização da população ocorre por meio dos algoritmos PF_NEH, GRASP_NEH e de modo aleatório. Os parâmetros do ED foram configurados conforme segue: *Np* = 10, *F* = 0,1 e *CR* = 0,1. O desempenho dos algoritmos de busca local populado com ED e GRASP são testados no *benchamark* de Taillard e comparados com o desempenho dos algoritmos HDDE, RAIS, IG, IG-reimplementada e MA. Os resultados mostraram que DE_PLS melhorou o resultado de 90 dos 120 problemas testados.

Tien et al. (2015) propõem um algoritmo de ED melhorado para o problema de minimização do *Makespan* em ambientes FSP. No algoritmo proposto um novo mecanismo de mutação é usado para ativar uma sequência apropriada para cada tarefa, portanto, o propósito central do estudo de Tien et al. (2015) é descobrir o esquema chave da melhor solução e tornar assimilado o operador no processo de mutação. No algoritmo proposto, a estratégia adotada é a *ED/best/1/bin* e a inicialização da população ocorre de modo aleatório. Os parâmetros do ED foram configurados conforme segue: Np = 100, F = {0,4, 0,8} e CR configurado no limiar. Para avaliar o desempenho da abordagem proposta, oito problemas testes do *Benchmark* de Taillard foram usados, e os resultados demonstram que a abordagem proposta apresenta melhor desempenho que o algoritmo EDA (*Estimation of Distribution Algorithm*).

Santucci, Baioletti e Milani (2016) introduzem uma abordagem algébrica original para algoritmos de evolução diferencial em espaços de busca combinatórios. Um operador de mutação diferencial algébrico para espaços combinatórios genéricos é definido, explorando o conceito de grupo finitamente gerado. O operador de mutação é especializado para o espaço de permutações por meio de um algoritmo *Bubble Sort* randomizado. Um algoritmo de ED Discreto, denominado DEP, para minimização do Tempo Total de Fluxo (*Total Flow Time*) em FSP é proposto. O algoritmo DEP contempla um operador de cruzamento para permutações (TPII - *Two-point crossover version II*), um novo esquema de seleção tendencioso, um procedimento memético de reinício, além de uma busca local, que após cada reinicialização é opcionalmente aplicada ao melhor indivíduo da população. A inicialização da população é feita por meio das heurísticas R_INIT e H_INIT, sendo Np = {50, 100, 200}. O algoritmo DEP não tem parâmetro de cruzamento CR e o parâmetro F é ajustado por meio de um esquema autoadaptativo. Os resultados fornecidos pelos algoritmos AGA, VNS4, GM-EDA, HGM-EDA, ILS e IGA são comparados com os resultados fornecidos pelo DEP para determinar seu desempenho. Os resultados experimentais mostram a superioridade do algoritmo proposto, que foi capaz de encontrar alguns novos limitantes para os problemas testados.

Shao e Pi (2016) propõem um algoritmo de ED autoguiado com busca em vizinhança, denominado NS-SGDE, para o problema de minimização do Tempo Máximo de Conclusão (*Máximo Completion Time*) em ambientes FSP. A

inicialização de uma parte da população do NS-SGDE é realizada através de um algoritmo DHS (*Discrete Harmonic Search*) com as heurísticas construtivas NEH, RAJ e FBR1 incorporadas e o restante da população é gerada de modo aleatório. Um agente guiado baseado em modelo probabilístico é proposto para guiar a fase de exploração do ED para gerar descendentes. Múltiplas operações de mutação e cruzamento baseado em agentes guiados são empregadas no algoritmo proposto para explorar mais efetivamente as soluções. Uma busca em vizinhança baseada em VNS (*Variable Neighborhood Search*) é projetada para melhorar a habilidade da busca. Após analises preliminares, os parâmetros do ED foram configurados conforme segue: Np = 100, F = 0,4 e CR = 0,3. A convergência do NS-SGDE é analisada de acordo com a teoria de Cadeias de Markov. Os resultados das experimentais computacionais mostraram que NS-SGDE proposto tem melhor desempenho que os algoritmos SGDE_NOHS, SGDE, EDA, DE, PSOMA, HBSA, HTLBO, ATPPSO, ODDE e ACGA.

As principais características dos trabalhos que tratam do desenvolvimento de métodos de solução baseados em abordagens de Evolução Diferencial para solução do problema de programação da produção em *Flow Shop* Permutacional são resumidas no Apêndice A.

4.2 ANÁLISES DAS PUBLICAÇÕES SOBRE EVOLUÇÃO DIFERENCIAL APLICADA AOS PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO EM *FLOW SHOP* PERMUTACIONAL

As análises das publicações sobre Evolução Diferencial Aplicada aos Problemas de Programação da Produção em *Flow Shop* Permutacional, no que diz respeito às características dos problemas foram efetuadas em relação a:

a) Tipo de função-objetivo: Monocritério, Bicritério e Multicritério;
b) Critério(s) de desempenho adotado(s);
c) Restrições adicionais consideradas.

As análises das publicações sobre Evolução Diferencial Aplicada aos Problemas de Programação da Produção em *Flow Shop* Permutacional, no que diz respeito às características dos algoritmos de ED foram efetuadas em relação a:

a) Estratégia de Evolução Diferencial;
b) Parâmetros de Controle: Determinístico, Adaptativo e Autoadaptativo;
c) Hibridizações do Algoritmo.

4.2.1 Tipo de Função Objetivo e Critérios de Desempenho Adotados

Na literatura orientada aos problemas de programação em ambientes *Flow Shop* Permutacional, 85% dos trabalhos (34 trabalhos) tratam do desenvolvimento de métodos de solução com função objetivo monocritério, 12,5% dos trabalhos (5 trabalhos) tratam do desenvolvimento de métodos de solução com função objetivo bicritério e 2,5% dos trabalhos (1 trabalho) desenvolve métodos monocritério e bicritério.

Nos estudos com funções monocritério, o *Makespan* e o Tempo de Fluxo (*Flow Time*) destacam-se como os critérios mais utilizados. O *Makespan* aparece em 73,52% dos trabalhos (25 trabalhos), enquanto o Tempo de Fluxo (*Flow Time*) é adotado em 26,47% dos trabalhos (9 trabalhos). O *Máximo Completion Time*, o Total *Tardiness*, o Máximo Lateness e o Número de Tarefas em Atraso são adotados como critério de desempenho em 2,94 % dos trabalhos (1 trabalho cada critério).

Nesta análise, destacam-se os trabalhos de Pan, Tasgetiren e Liang (2008), Lin e Yin (2013), Tasgetiren et al. (2013) que investigam o problema com função monocritério, porém na experimentação computacional utilizam dois critérios de desempenho. Destaca-se também o trabalho de Zheng e Yamashiro (2010), que investiga o problema com função monocritério, porém na experimentação computacional utiliza três critérios de desempenho.

Dos estudos com funções bicritério, destacam-se os critérios *Makespan* e Máximo *Tardiness*, que são considerados em 60% dos trabalhos (3 trabalhos). Os critérios *Makespan* e Total *Flow Time* são considerados em 20 % dos trabalhos (1 trabalho) e os critérios *Makespan* e Custo dos Recursos também são adotados em 20 % dos trabalhos (1 trabalho).

No trabalho que propõem métodos monocritério e bicritério, apresentado por Qian et al. (2008), para o caso monocritério, o critério de desempenho adotado foi o *Makespan*, e para o caso bicritério, a função objetivo foi composta pelos critérios de *Makespan* e Máximo *Tardiness*.

4.2.2 Restrições Adicionais Consideradas

Dos estudos que tratam dos problemas de programação em ambientes *Flow Shop* Permutacional, 52,5% (21 trabalhos) apresenta algum tipo de restrição adicional. Ressalta-se que vários trabalhos incorporam mais de um tipo de restrição adicional no problema de programação em ambientes *Flow Shop* Permutacional.

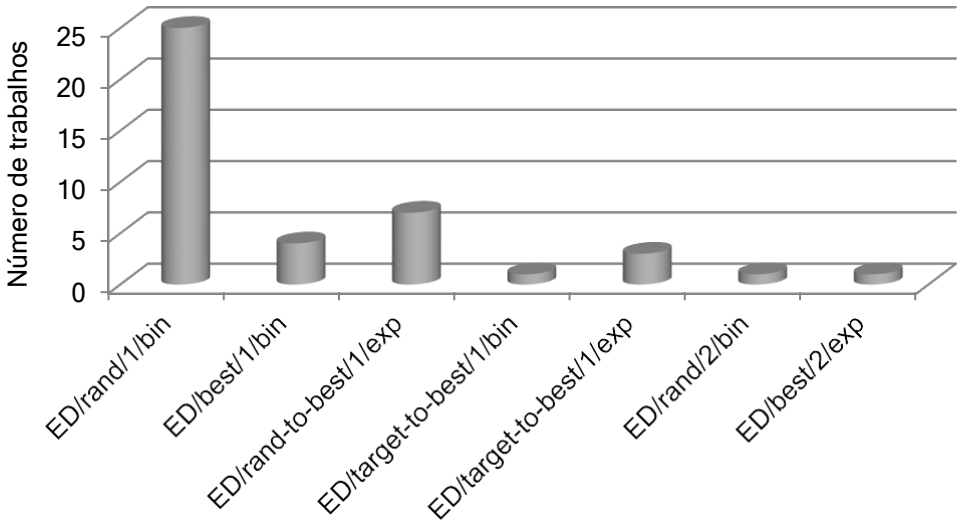
Dos trabalhos que apresentam restrições adicionais, destacam-se as restrições de *No-Wait* que aparecem em 52,38 % dos trabalhos (11 trabalhos). Restrições de *no-idle*, *setups* dependentes da sequencia e diferentes datas de liberação são verificadas em 14,28% dos trabalhos (3 trabalhos cada). Restrições de *buffers* limitados aparecem em 9,52% (2 trabalhos), enquanto que restrições de bloqueio, fluxo reentrante tamanhos de lotes, horizonte de tempo e tempos de processamento dependentes dos recursos são restrições consideradas em 4,76 % dos trabalhos (1 trabalho cada).

Nesta análise, destaque deve ser dado aos trabalhos de Hu et al. (2012), Lobato, Gedraite e Neiro (2012) e Qian et al. (2012a) que incorporam dois ou mais tipos de restrições adicionais ao problema de programação investigado.

4.2.3 Estratégia de Evolução Diferencial e Parâmetros de Controle

Em relação às estratégias de ED utilizadas nos trabalhos investigados, verificou-se a predominância das estratégias *ED/rand/1/bin* e *ED/rand-to-best/1/exp*. O número de trabalhos de acordo com a estratégia de ED adotada é apresentado no Gráfico 4.2.

Gráfico 4.2 - Número de trabalhos de acordo com as estratégias de ED adotadas nos algoritmos para *Flow Shop* Permutacional



Fonte: o autor, 2017.

Ainda em relação às estratégias de ED adotadas nos estudos orientados ao *Flow Shop* Permutacional, destacam-se os trabalhos de Guimarães et al. (2013) e Amirian e Sahraeian (2014). Guimarães et al. (2013) e Amirian e Sahraeian (2014) analisam, na experimentação computacional, duas diferentes estratégias de ED, enquanto Tasgetiren et al. (2010) e Santuci, Baioletti e Milani (2016) propõem procedimentos diferentes para a geração dos vetores mutantes.

Quanto aos parâmetros de controle, de acordo com a classificação dos mecanismos de ajuste dos parâmetros de controle, apresentada no capitulo 3, seção 3.3, subseção 3.3.1, verificou-se que em 90% dos trabalhos (36 trabalhos) identificados para programação em ambientes *Flow Shop* Permutacional, os mecanismos de ajuste dos parâmetros de controle são determinísticos, ou seja, fixados empiricamente pelos autores com base na literatura especializada.

Somente em 10% dos trabalhos (4 trabalhos) os mecanismos de ajuste dos parâmetros são autoadaptativos. No estudo de Xu, Xiang e Wang (2010) os parâmetros *F* e *CR* são ajustados dinamicamente para cada indivíduo durante o processo evolutivo. Já no estudo de Li e Yin (2013) somente o parâmetro *CR* é ajustado autoadaptativamente, enquanto que nos estudos de Santuci, Baioletti e Milani (2014) e Santuci, Baioletti e Milani (2016) somente o parâmetro *F* é ajustado autoadaptativamente.

4.2.4 Hibridizações

Em relação a hibridização de métodos no procedimento de geração da população inicial, pode ser verificado que em 50% dos trabalhos (20 trabalhos) a inicialização da população é aleatória e que em 47.5% dos trabalhos (19 trabalhos) a inicialização de uma parcela da população ocorre por meio de diferentes regras de prioridade e métodos heurísticos e o restante ocorre de modo aleatório.

Ainda em relação à inicialização da população, destaca-se aqui o trabalho de Zheng e Yamashiro (2010) em que a inicialização da população ocorre no Algoritmo Evolucionário Quântico (*Quantum Evolutionary Algorithm*) e o algoritmo de ED é aplicado a cada *Q-bit* para todos os cromossomos quantum gerados na fase de inicialização.

Nos trabalhos analisados foi verificada a utilização das regras de prioridade SPT, LPT, SOT e LOT e dos métodos heurísticos NN, NEH, PWQ, Insertion Improvement, NDY, SPT/TWKR, F_NEH, R_INIT, H_INIT, DHS, PFBR1, RAJ, Busca Local baseada em Inserção, LR(*n*/*m*).

O emprego de mecanismos adicionais de busca, visando evitar estagnações e/ou melhorar a capacidade de exploração do algoritmo no espaço de soluções constitui uma característica que tem se destacado no desenvolvimento de algoritmos de ED para os problemas de programação da produção. Neste contexto, verifica-se que 90% dos trabalhos (36 trabalhos) incorporam mecanismos de busca local ou algoritmos baseados em busca local aos algoritmos de ED desenvolvidos.

Foram verificadas hibridizações dos algoritmos de ED com os algoritmos VNS (*Variable Neighborhood Search*), VND (*Variable Neighborhood Descendent*), VPS (*Variable Parameter Search*), VIG (*Variable Iterated Greedy*), RLS (*Referenced Local Search*), IG (*Iterated Greedy*), ILS (*Iterated Local Search*), IIS-BLS (*Individual Improving Scheme- Based Local Search*), *Greedy-Based Local Search*, GRASP (*Greedy Randomized Adaptive Search Procedure*), SA (*Simulated Annealing*), NEH (*Nawaz, Enscore and Ham*), TS (*Tabu Search*) e NSGA-II (*Non-dominated Sorting Genetic Algorithm* II).

Capítulo 5 - ABORDAGENS DE EVOLUÇÃO DIFERENCIAL PROPOSTA PARA PROGRAMAÇÃO DA PRODUÇÃO EM *FLOW SHOP* PERMUTACIONAL

Neste capítulo são apresentadas uma variante do algoritmo de Evolução Diferencial Discreta (EDD) e duas variantes do algoritmo de Evolução Diferencial Discreta Autoadaptativa (EDDA), desenvolvidas para a minimização do *Makespan* em problemas de programação da produção em ambientes *Flow Shop* Permutacional.

5.1 EVOLUÇÃO DIFERENCIAL DISCRETA

Nesta tese, a versão clássica do algoritmo de ED (STORN; PRICE, 1995; STORN; PRICE, 1997) disponível em *International Computer Science Institute* - ICSI (2017) foi adaptada para o caso de minimização do *Makespan* em programação da produção em *Flow Shop* Permutacional e denominada Evolução Diferencial Discreta para Flow Shop Permutacional (EDD-FSP).

5.1.1 Algoritmo EDD-FSP

No algoritmo EDD-FSP a população inicial de vetores é formada aleatoriamente e uniformemente distribuída entre os limites superiores e inferiores APLICADA algoritmo um vetor $x = \{x_1, x_2, ..., x_n\}$ é aleatoriamente produzido de acordo com a equação (5.1), com x_{min} = -1,0 e x_{max} = 1,0 para cada dimensão *j*.

$$x_{j,i,0} = rand_j(0,1) * (x_{min} - x_{max}) + x_{max} \quad (5.1)$$

onde $rand_j$ é a função geradora de números aleatórios uniformemente distribuídos no intervalo 0 e 1 e o subscrito *j* indica que a cada parâmetro é gerado um valor aleatório novo.

No algoritmo EDD-FSP a regra LOV (*Largest-Order-Value*) proposta por Qian e Wang (2008) é aplicada para converter os vetores no domínio contínuo

$x_i = \{x_{i,1}, x_{i,2},..., x_{i,n}\}$ em vetores permutações de tarefas no domínio discreto $\pi_i = \{\pi_{i,1}, \pi_{i,2},..., \pi_{i,n}\}$.

De acordo com regra LOV os vetores contínuos $xi = \{x_{i,1}, x_{i,2},..., x_{i,n}\}$ são primeiramente classificados por ordem descendente para obter uma sequência $\varphi_i = \{\varphi_{i,1}, \varphi_{i,2},..., \varphi_{i,n}\}$. Então uma permutação de tarefas πé calculada utilizando a equação (5.2) tal que:

$$\pi_{i,\varphi_{i,k}} = k \tag{5.2}$$

onde a dimensão *k* varia de 1 a *n*.

No Apêndice B, um exemplo simples, extraído de Qian et al. (2008), ilustra a aplicação da regra LOV.

No algoritmo EDD-FSP, do mesmo modo que na versão clássica do algoritmo de ED, a estratégia *ED/rand/1/bin* proposta por Storn e Price (1995, 1997) é utilizada para geração de um vetor mutante.

Na estratégia *ED/rand/1/bin*, para cada vetor alvo $x_{i,G}, i = 0,1,.., N_p - 1$em uma geração G, um vetor mutante $v_{i,G}$, resultado da combinação de três diferentes vetores, escolhidos aleatoriamente, é gerado utilizando a equação (5.3):

$$v_{i,G} = x_{r0,G} + F(x_{r1,G} - x_{r2,G}) \tag{5.3}$$

onde $r_0, r_1, r_2 \in \{1,2,..., Np\}$ são índices gerados aleatoriamente com distribuição uniforme e representados por inteiros, mutuamente diferentes e $F \in [0, 2]$ é um fator constante e real que controla a amplificação da variação diferencial.

No algoritmo EDD-FSP, o fator *F* é configurado pelo usuário no início do processo de otimização e não sofre alteração no decorrer do processo evolutivo.

Após a mutação, no algoritmo EDD-FSP uma operação de cruzamento binário é aplicada para formar o vetor teste $u_{i,G}$, conforme equação (5.4).

$$u_{i,G} = u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{se}(rand_j[0,1] \le CR_i) \text{ ou } j = j_{rand} \\ x_{j,i,G} & \text{casocontrário} \end{cases} \tag{5.4}$$

onde $rand_j(a,b)$ é um número gerado com distribuição uniforme no intervalo (*a*,*b*] e novamente gerado para cada *j*, j_{rand} = $randint_i(1, D)$ é um inteiro escolhido aleatoriamente com distribuição uniforme no intervalo de 1 a *D* e novamente gerado

para cada *i* e a probabilidade $CR_i \in (0, 1]$ que corresponde à fração média de componentes de vetores que são herdados do vetor de mutação.

No algoritmo EDD-FSP, o fator *CR* é configurado pelo usuário no início do processo de otimização e não sofre alteração no decorrer do processo evolutivo.

No algoritmo EDD-FSP, para selecionar os membros da próxima geração, o vetor de ensaio (vetor teste) $u_{i,G}$ somente irá substituir o vetor alvo $v_{i,G}$ se produzir um valor menor do que a função custo do vetor alvo, ou seja, se apresentar um valor menor para a duração total da programação ou *Makespan*. Ou seja, se o vetor $u_{i,G}$ produz um valor menor do que a função de custo de $x_{i,G}$, então $x_{i,G+1}$ é definido como $u_{i,G+1}$, caso contrário, o antigo valor de $x_{i,G}$ é retido, conforme equação (5.5), onde

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{se } f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} & \text{casocontrário} \end{cases} \qquad (5.5)$$

O pseudocódigo do algoritmo EDD-FSP é apresentado no Quadro 5.1 a seguir.

Quadro 5.1 - Pseudocódigo do algoritmo EDD-FSP

Pseudocódigo do Algoritmo EDD-FSP

Início
Faça G = 0
Inicie aleatoriamente uma população de *NP* indivíduos *x*
Aplique a regra LOV a população inicial e avalie a aptidão de todos os indivíduos
  Para *g* = 1 a *G*
    Para *i* = 1 a *NP*
      Selecione aleatoriamente com distribuição uniforme $r_0, r_1, r_2 \in \{1,2,..., Np\}$
      Gere j_{rand} = randint(1, *D*)...
      Para $j = 1$ a *D*
        Se *j* = j_{rand} ou rand(0,1) < CR_i
        $u_{j,i,g} = v_{j,i,g} = x_{r0,G} + F(x_{r1,G} - x_{r2,G})$

        Senão
          $u_{j,i,g} = x_{j,i,g}$
        Fim //Se
      Fim //Para
      Aplique a regra LOV a nova população e avalie a aptidão de todos os membros
      Se $f(u_{i,g}) \leq f(x_{i,g})$
        $x_{i,g+1} = x_{i,g}$
      Senão
        $x_{i,g+1} = u_{i,g}$;
      Fim //Se
    Fim //Para
  Fim // Para
Fim

Fonte: o autor, 2017.

5.2 EVOLUÇÃO DIFERENCIAL DISCRETA AUTOADAPTATIVA

As abordagens autoadaptativas propostas nesta tese baseiam-se em conceitos de autoadaptação dos parâmetros de controle do algoritmo, dado que, o desempenho de algoritmos baseados em abordagens de ED ainda é dependente da configuração dos parâmetros de controle, tais como fator de mutação (*F*), taxa de cruzamento (*CR*) e tamanho da população (*Np*) (PRICE; STORN; LAMPINEN, 2005; ZHANG; SANDERSON, 2007; ZHANG; SANDERSON, 2009a; FAN; YAN, 2015; ZAMUDA; BREST, 2015; FAN; YAN, 2016; BANITALEBI; AZIS; AZIS, 2016; FAN; ZHANG, 2016; PIEREZAN et al., 2017).

A escolha apropriada dos parâmetros de controle do algoritmo depende da função a ser minimizada e da variante do algoritmo de ED utilizada no processo de

otimização. Quando as propriedades dos problemas de otimização são desconhecidas é difícil escolher parâmetros de controle adequados, sendo nestes casos, geralmente utilizado o método de tentativa e erro, que requer tediosos testes de otimização, para obter um ajuste adequado dos parâmetros do algoritmo (ZHANG; SANDERSON, 2009a; FAN; ZHANG, 2016).

Conforme já mencionado, várias diretrizes empíricas para selecionar os parâmetros de controle e as estratégias para melhorar o desempenho do algoritmo de ED são apresentadas na literatura. No entanto, essas diretrizes geralmente carecem de justificativas suficientes, pois se baseiam em experiências específicas (FAN; ZHANG, 2016).

Para evitar a necessidade de ajustes de parâmetros dependentes do problema, segundo Peñuñuri et al. (2016) uma possível solução para o problema de encontrar um bom conjunto de parâmetros de controle consiste em implementar um algoritmo que mantenha os elementos essenciais do algoritmo de ED, mas é capaz de adaptar os parâmetros de controle durante o processo evolucionário.

Além de uma configuração adequada dos parâmetros de controle, um compromisso adequado entre *exploration* e *exploitation* é relevante para aumentar a precisão e a eficiência de um algoritmo estocástico de busca. Consequentemente, o algoritmo deve ser capaz de obter um compromisso adequado entre explorar o espaço de busca o máximo possível para aumentar a probabilidade de encontrar o ótimo global (*exploration*) e explorar a vizinhança de pontos do espaço de busca já visitados anteriormente de forma a aumentar a velocidade de convergência do processo de busca (*exploitation*) (CREPINSEK; LIU; MERNIK, 2013).

Os parâmetros de controle adaptativos ou autoadaptativos, se bem projetados podem melhorar a robustez de um algoritmo através da adaptação dinâmica dos parâmetros para as características de diferentes problemas, sem necessidade de tentativa e erro. Em adição, a taxa de convergência pode ser melhorada se os parâmetros de controle são adaptados a valores apropriados em diferentes estágios de evolução de um problema específico (ZHANG; SANDERSON, 2009a).

Com base nestas considerações, o objetivo das abordagens propostas nesta tese é a implementação de algoritmos que sejam capazes de cumprir os seguintes pontos:

- Garantir equilíbrio adequado entre *exploration* e *exploitation*, ou seja, capaz de explorar cada região do espaço de busca (*exploration*) ao mesmo tempo em que é capaz de obter soluções ainda melhores a partir de um conjunto de boas soluções a cada geração (*exploitation*); e
- Capaz de adaptar os valores dos parâmetros de controle durante o processo de otimização, o que é crucial para aumentar a velocidade de convergência e reduzir o risco de estagnação durante o processo de otimização.

Os algoritmos aqui desenvolvidos tiveram como base o algoritmo JADE (*J Adaptive Differential Evolution*) proposto por Zhang e Sanderson (2007) e Zhang e Sanderson (2009b) e o algoritmo SaDE (*Self-adaptive Differential Evolution*) proposto por Qin e Suganthan (2005) e Qin, Huang e Suganthan (2009).

Os algoritmos de Evolução Diferencial Discreta Autoadaptativa (EDDA) para *Flow Shop* Permutacional (FSP) desenvolvidos neste trabalho foram denominados Evolução Diferencial Discreta Autoadaptativa para *Flow Shop* Permutacional (EDDA-FSP_1 e EDDA-FSP_2). Cada uma destas versões será descrita em detalhes nas próximas seções.

A avaliação e análise do desempenho e eficiência de cada uma das versões serão apresentadas e discutidas no Capítulo 6.

5.2.1 Algoritmo EDDA-FSP_1

O primeiro algoritmo autoadaptativo proposto neste trabalho, denominado EDDA-FSP_1, foi desenvolvido tendo como base os conceitos autoadaptativos e uma nova estratégia de mutação ED/*current-to-p-best* com arquivo externo presentes no algoritmo JADE (*J Adaptive Differential Evolution*). A estratégia de mutação ED/*current-to-p-best* com arquivo externo e a autoadaptação dos parâmetros μ_{CR} e μ_F utilizados para gerar o fator de mutação F_i e a taxa de cruzamento CR_i associados com cada indivíduo $x_{i,G}$ visam melhorar a taxa de convergência e a confiabilidade do desempenho do algoritmo (ZHANG e SANDERSON, 2007; ZHANG e SANDERSON, 2009b).

No algoritmo EDDA-FSP_1 a população inicial de vetores é formada do mesmo modo que nos algoritmos EDD-FSP e assim como no algoritmo EDD-FSP, a regra

LOV (*Largest-Order-Value*) é aplicada para converter os vetores contínuos em vetores permutações de tarefas.

A estratégia de ED adotada no algoritmo EDDA-FSP$_1$ é a ED/*current-to-p-best/1/bin* com arquivo externo. Na estratégia ED/*current-to-p-best/1/bin*, que é uma generalização da estratégia *ED/current-to-best/1/bin*, qualquer uma das 100*p*% soluções podem ser aleatoriamente escolhidas com distribuição uniforme para desempenhar o papel projetado exclusivamente para a única melhor solução na *ED/current-to-best/1/bin* (ZHANG; SANDERSON, 2007).

Na estratégia *ED/current-to-p-best/1/bin* com arquivo externo, para cada vetor alvo $x_{i,G}, i=1,2,...,Np$, da população atual, um vetor mutante é gerado de acordo com a equação (5.6).

$$v_{i,G} = x_{i,G} + F_i(x^p_{best} - x_{i,G}) + F_i(x_{r1,G} - \tilde{x}_{r2,G}) \tag{5.6}$$

onde $x_{i,G}$, $x_{r1,G}$ e x^p_{best} são selecionados de *P*, enquanto $\tilde{x}_{r2,G}$ é selecionado aleatoriamente da união, $P \cup A$, sendo que *A* denota o conjunto de soluções inferiores arquivadas, enquanto *P* denota a população corrente.

No algoritmo EDDA-FSP$_1$ o arquivo *A* inicia vazio, e então ao final de cada geração, os vetores alvos, que foram substituídos pelos respectivos vetores experimentais durante a seleção, são adicionados ao arquivo. Se o tamanho do arquivo exceder um determinado limite, algumas soluções são eliminadas de forma aleatória mantendo-se, portanto, o tamanho do arquivo fixo e dentro do limite estipulado.

No algoritmo EDDA-FSP$_1$ após o processo de mutação, o cruzamento binário também é empregado e o processo de seleção ocorre de modo idêntico ao processo de seleção empregado nos algoritmos EDD-FSP.

No algoritmo EDDA-FSP$_1$ a autoadaptação é aplicada aos parâmetros μ_{CR} e também μ_F, os quais são utilizados para gerar o fator de mutação F_i e a taxa de cruzamento CR_i associados com cada indivíduo $x_{i,G}$. F_i e CR_i são então usados para gerar o vetor teste $u_{i,G}$ quando $x_{i,G}$ serve como vetor base em (5.4) e (5.5).

Em cada geração *G,* uma probabilidade de cruzamento CR_i para cada indivíduo $x_{i,G}$ é independentemente gerada, de acordo com uma distribuição normal, com média μ_{CR} e desvio padrão 0,1, respectivamente, e então truncado para [0, 1].

$$CR_{i,} = randn_i(\mu_{CR}, 0,1) \tag{5.7}$$

Similarmente, em cada geração *G,* o fator de mutação F_i para cada indivíduo $x_{i,G}$, é independentemente gerado, de acordo com uma distribuição de Cauchy (ZHANG; SANDERSON, 2009b), com média μ_F e desvio padrão 0,1.

$$F_{i,} = randc_i(\mu_F, 0,1) \tag{5.8}$$

Os parâmetros μ_{CR} e μ_F são iniciados em 0,5 no início do processo de otimização e atualizados ao final de cada geração, de acordo com as equações (5.9) e (5.10), respectivamente, tal que:

$$\mu_F = (1-c).\mu_F + média_L(S_F) \tag{5.9}$$

$$\mu_{CR} = (1-c).\mu_{CR} + média_A(S_{CR}) \tag{5.10}$$

Nas equações (5.9) e (5.10), *c* é uma constante real pertencente ao intervalo [0,1], S_F é o conjunto de todos os fatores de mutação bem sucedidos e S_{CR} é o conjunto de todas as taxas de cruzamento bem sucedidas na geração de $G.F_i$ e CR_i são considerados bem sucedidos se o vetor experimental $u_{i,G}$, gerado utilizando-se estes parâmetros de controle, for melhor que seu respectivo vetor alvo $x_{i,G}$.

A $média_A(.)$ corresponde a média aritmética e a $média_L(.)$ corresponde à média Lehmer (ZHANG; SANDERSON, 2009b), calculada por meio da equação (5.11).

$$\text{média}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{5.11}$$

De acordo com Zhang e Sanderson (2007) a adaptação de μ_{CR} baseia-se no princípio de que os melhores valores dos parâmetros de controle tendem a gerar indivíduos que são mais propensos a sobreviver e, portanto, esses valores devem ser propagados para as gerações seguintes. A adaptação de μ_F atribui maior peso nos fatores de mutação bem sucedidos por meio do uso da média Lehmer na equação (5.10), em vez da média aritmética, como usada na adaptação de μ_{CR}.

Ao contrário do algoritmo de Evolução Diferencial Clássico, em que os parâmetros *F* e *CR* necessitam ser ajustados pelo usuário, uma vez que cada problema específico requer uma configuração apropriada, é esperado que os dois novos parâmetros *c* e *p*, presentes no algoritmo EDDA-FSP$_1$, sejam insensíveis a diferentes problemas, como ocorre no algoritmo JADE. De acordo com Zhang e Sanderson (2009a) o *c* controla a taxa de adaptação dos parâmetros e *p* determina a ganancia da estratégia de mutação.

O pseudocódigo do algoritmo EDDA-FSP$_1$ é apresentado no Quadro 5.2 a seguir.

Quadro 5.2 - Pseudocódigo do algoritmo EDDA-FSP$_1$

Pseudocódigo do Algoritmo EDDA-FSP$_1$

Início
Faça $G = 0$
Inicie aleatoriamente uma população de *NP* indivíduos
Inicialize $\mu_{CR} = 0{,}5$; $\mu_F = 0{,}5$; $A = \varphi$.
Aplique a regra LOV a população inicial e avalie a aptidão de todos os indivíduos
  Para $g = 1$ a G
    $S_F = \varphi$; $S_{CR} = \varphi$;
    Para $i = 1$ a NP
      Gere $CR_i = \text{randn}(\mu_{CR}, 0{,}1)$, $F_i = \text{randc}_i(\mu_F, 0{,}1)$
      Selecione aleatoriamente, segundo uma distribuição uniforme $x^p_{best,g}$ como um dos $100p\%$ melhores vetores
      Selecione aleatoriamente de $x_{r1,g} \neq x_{i,g}$ da população corrente P
      Selecione aleatoriamente com distribuição uniforme $x_{r2,g} \neq x_{r1,g}$ de $P \cup A$
      $v_{i,g} = x_{i,g} + F_i(x^p_{best,g} - x_{i,g}) + F_i(x_{r1,g} - \tilde{x}_{r2,g})$
      Gere j_{rand} = randint(1, D)
        Para $j = 1$ a D
          Se $j = j_{rand}$ ou rand(0,1) < CR_i
            $u_{j,i,g} = v_{j,i,g}$
          Senão
            $u_{j,i,g} = x_{j,i,g}$
          Fim //Se
        Fim //Para
      Aplique a regra LOV a nova população e avalie a aptidão de todos os indivíduos
        Se $f(u_{i,g}) \leq f(x_{i,g})$
          $x_{i,g+1} = x_{i,g}$
        Senão
          $x_{i,g+1} = u_{i,g}$; $x_{i,g} \rightarrow A$; $CR_i \rightarrow S_{CR}$; $F_i \rightarrow S_F$
        Fim //Se
    Fim //Para
    Aleatoriamente remova soluções de A de modo que $|A| \leq NP$
    $\mu_{CR} = (1-c).\mu_{CR} + c.\text{media}_A(S_{CR})$
    $\mu_F = (1-c).\mu_F + c.\text{media}_L(S_F)$
  Fim //Para
Fim

Fonte: o autor, 2017.

5.2.2 Algoritmo EDDA-FSP$_2$

O segundo algoritmo autoadaptativo proposto neste trabalho, denominado EDDA-FSP$_2$, foi desenvolvido tendo como base o algoritmo SaDE (*Self-adaptive Differential Evolution*) proposto por Qin e Suganthan (2005) e estendido por Qin, Huang e Suganthan (2009), em que a escolha da estratégia de ED e dois dos parâmetros de controle *F* e *CR* não precisam ser pré-especificados.

No algoritmo EDDA-FSP$_2$ durante a evolução, a estratégia de ED (estratégias de geração de vetor experimental) e os parâmetros de controle *F* e *CR* são gradualmente autoadaptados de acordo com a experiência de aprendizagem.

No algoritmo SaDE, cada indivíduo tem seus próprios parâmetros de *F* e *CR* e um par de parâmetros de controle auxiliares ajudam a atualizar esses parâmetros dinamicamente. Um mecanismo probabilístico projetado para selecionar de forma adaptativa uma estratégia de pesquisa mais produtiva e um método concebido para adaptar a taxa de cruzamento com base na medição estatística de alguns dos melhores parâmetros da taxa de cruzamento também estão presentes no algoritmo SaDE (BANITALEBI; AZIZ; AZIS, 2016).

Assim como na SaDE, no algoritmo EDDA-FSP$_2$ a ideia da adaptação estratégia é selecionar probabilisticamente uma das várias estratégias disponíveis e aplicar à população atual.

No algoritmo EDDA-FSP$_2$ a população inicial de vetores é formada do mesmo modo que nos algoritmos EDD-FSP e EDDA-FSP$_1$ e assim como nos algoritmos EDD-FSP e EDDA-FSP$_1$ a regra LOV (*Largest-Order-Value*) é aplicada para converter os vetores contínuos em vetores permutações de tarefas.

No algoritmo EDDA-FSP$_2$ as estratégias de ED são escolhidas para serem aplicadas a indivíduos na população corrente com probabilidades proporcionais às suas taxas de sucesso anteriores para gerar potencialmente boas novas soluções.

Tendo como base Qin e Suganthan (2009) as estratégias selecionadas, devido ao seu bom desempenho em problemas com diferentes características, foram: ED/*rand*/1/bin, ED/*rand*/2/bin, ED/*rand-to-best*/2/bin e ED/*current-to-rand*/1.

Nas estratégias ED/*rand*/1/bin e ED/*rand*/2/bin para cada vetor alvo $x_{i,G}, i = 1,2,..., Np$, da população atual, um vetor mutante é gerado de acordo com as equações (5.12) e (5.13), respectivamente.

$$v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2G}) \tag{5.12}$$

$$v_{i,G} = x_{r0,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G}) \tag{5.13}$$

Nas estratégias ED/*rand-to-best*/2/bin e ED/*current-to-rand*/1 para cada vetor alvo $x_{i,G}, i = 1,2,...,Np$, da população atual, um vetor mutante é gerado de acordo com as equações (5.14) e (5.15), respectivamente.

$$v_{i,G} = x_{i,G} + F.(x_{best,G} - x_{i,G}) + F.(x_{r0,G} - x_{r1,G}) \tag{5.14}$$

$$v_{i,G} = x_{i,G} + K.(x_{r1,G} - x_{i,G}) + F.(x_{r2,G} - x_{r3,G}) \tag{5.15}$$

Na equação (5.15) *K* é um valor escolhido aleatoriamente dentro do intervalo [0,1].

No algoritmo EDDA-FSP$_2$ após o processo de mutação, o cruzamento binário também é empregado e o processo de seleção ocorre de modo idêntico ao processo de seleção empregado nos algoritmos EDD-FSP e EDDA-FSP$_1$.

No algoritmo EDDA-FSP$_2$, a cada geração *G* uma estratégia é escolhida aleatoriamente para cada vetor alvo de acordo com uma probabilidade $p_{k,G}, (k = 1,2,3,4)$, onde $p_{k,G}$ representa a probabilidade da *k*-ésima estratégia ser escolhida na geração *G.* Cada uma das probabilidades $p_{k,G}, (k = 1,2,3,4)$ é inicializada a 0,25 e, então, atualizada durante o processo de otimização por meio da equação 5.16.

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{4} S_{k,G}} \tag{5.16}$$

Tal que $S_{k,G}$ é obtido pela equação 5.17.

$$S_{k,g} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \varepsilon \tag{5.17}$$

onde $ns_{k,g}$ é o número de vetores experimentais bem sucedidos na geração *G* gerados pela *k*-ésima estratégia $nf_{k,g}$ representa o número de vetores experimentais gerados pela *k*-ésima estratégia e descartados durante o processo de seleção na geração *G*. O parâmetro ε é uma constante real não nula utilizada para evitar uma probabilidade nula caso $ns_{k,g} = 0$. *LP* é o período de aprendizagem.

No algoritmo EDDA-FSP$_2$, *F* e *CR* também são adaptativamente alterados durante o processo de evolução. A cada geração *G*, um fator de mutação F_i e uma taxa de cruzamento CR_i são gerados aleatoriamente para cada vetor alvo $x_{i,G}$ a partir de uma distribuição normal. F_i é gerado a partir de uma distribuição normal com média de 0,5 e desvio padrão de 0,3, enquanto CR_i é gerado a partir de uma distribuição normal com média $CRm_k, (k=1,2,3,4)$ e desvio padrão de 0,1:

$$F_i \sim N(0{,}5;0{,}3), i=1,2,...,Np \tag{5.18}$$

$$CR_i \sim N(CRm_k;0{,}1), i=1,2,...,Np \tag{5.19}$$

Um valor médio CRm_k é associado a cada uma das estratégias de geração de vetores experimentais $(k=1,2,3,4)$. Dessa forma, o valor médio a ser utilizado para o cálculo de CR_i dependerá da estratégia de geração de vetor experimental associado a $x_{i,G}$.

Os parâmetros CRm_k são inicializados a 0,5 e, em seguida, com base nos valores de taxa de cruzamento que geraram vetores experimentais bem sucedidos durante o período de aprendizagem. Basicamente, ao final de cada geração, CRm_k é atualizado com o valor da mediana do conjunto formado pelos fatores de mutação bem sucedidos nas últimas *LP* (período de aprendizagem) gerações associados a *k*-ésima estratégia.

O pseudocódigo do algoritmo EDDA-FSP$_2$ é apresentado no Quadro 5.3 a seguir.

Quadro 5.3 - Pseudocódigo do algoritmo EDDA-FSP$_2$

Pseudocódigo do Algoritmo EDDA-FSP$_2$

Início
Faça G = 0 e inicialize aleatoriamente uma população de *NP* indivíduos
Inicialize CR_{mk} ; $p_{k,G}$; e LP
Aplique a regra LOV a população inicial e avalie a aptidão de todos os indivíduos
  Enquanto critério de parada não satisfeito Faça
    Calcule a probabilidade da estratégia e atualize as memórias de sucesso e de fracasso
    Se *G* > *LP*
      Para *k* = 1 a *K*
        Atualize $p_{k,G}$ e remova $ns_{k,G-LP}$ e $nf_{k,G-LP}$ das memórias de sucesso e de fracasso
      Fim// Para
    Fim// Se
    Designe uma estratégia para geração do vetor teste e parâmetros para cada vetor alvo
    Para *i* = 1 a *NP*
      $F_i = \text{Normrnd}(0{,}5, 0{,}3)$
    Fim //Para
    Se $G \geq LP$
      Para *k* = 1 a *K*
        $CR_{m,k} = \text{Média}(\text{CRMemória}_k)$
      Fim //Para
    Fim // Se
    Para *k* = 1 a *K*
      Para *i* = 1 a *NP*
        $CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$
        Enquanto $CR_{m,i} < 0$ ou $CR_{m,i} > 1$ Faça $CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$
      Fim //Para
    Fim //Para
    Gere uma nova população de acordo com a estratégia *k* e os parâmetros F_i e CR_i associados a geração do vetor teste e reinicie aleatoriamente o vetor teste dentro do espaço de busca se qualquer variável estiver fora dos limitantes.
    Aplique a regra LOV a população inicial e avalie a aptidão de todos os indivíduos
    Para *i* = 1 a *NP*
      Se $f(u^k_{i,g}) \leq f(X_{i,g})$
        $X_{i,g+1} = u^k_{i,g}$ $f(X_{i,g+1}) = u^k_{i,g}$
        $ns_{k,G} = ns_{k,G} + 1$
        Armazene $CR_{k,i}$ em CRMemória_k
        Se $f(u_{i,g})\ f(X_{best,g})$
          $X_{best,g} = u_{i,g}$, $f(X_{best}) = f(u_{i,g})$
        Fim //Se
          Senão $nf_{i,g} = nf_{i,g} + 1$
        Fim //Se
    Fim //Para
    Armazene $ns_{k,G}$ e $nf_{k,}$ nas memórias de sucesso e de fracasso
    Faça G = G + 1
Fim

Fonte: o autor, 2017.

Capítulo 6 DELINEAMENTO DOS EXPERIMENTOS COMPUTACIONAIS

Este capítulo tem como propósito apresentar o delineamento da experimentação computacional realizada nesta tese. Neste capítulo são explicitados os casos selecionados para testar os algoritmos propostos, bem os procedimentos para geração e análise dos dados.

6.1 CASOS SELECIONADOS PARA REALIZAÇÃO DOSTESTES

Os algoritmos de Evolução Diferencial propostos foram testados em 110 casos, disponíveis nos *Benchmarks* de Carlier (1978), Heller (1960), Reeves (1995) e Taillard (1993).

Os *Benchmarks* foram obtidos em *Operations Research Library* (OR *Library*) e LIFL (*Laboratoire d'Informatique Fondamentale de Lille*). Os valores para o *Makespan* (C_{max}) que caracterizam os limitantes superiores dos casos foram obtidos *Operations Research Library* (OR *Library*), Mokhtari, Abadi e Cheraghalikhani (2011) e Ancău (2012).

Nas Tabelas 6.1 a 6.4 são apresentados os casos selecionados para testes dos *Benchmarks* de Carlier, Heller, Reeves e Taillard, respectivamente.

Tabela 6.1 – Casos selecionados para testes: *Benchmark* de Carlier

Caso	Descrição	Número de Tarefas	Número de Máquinas	C_{max}
1	Car_001_11_05	11	5	7038
2	Car_002_13_04	13	4	7166
3	Car_003_12_05	12	5	7312
4	Car_004_14_04	14	4	8003
5	Car_005_10_06	10	6	7720
6	Car_006_08_09	8	9	8505
7	Car_007_07_07	7	7	6590
8	Car_008_08_08	8	8	8366

Fonte: o autor, 2017.

Os tempos de processamento das tarefas nas máquinas dos casos da Tabela 6.1 (*Benchmark* de Carlier) são detalhados no Apêndice C.

Tabela 6.2 – Casos selecionados para testes: *Benchmark* de Heller

Caso	Problema Teste	Número de Tarefas	Número de Máquinas	C_{max}
9	Hel_001_20_10	20	10	136
10	Hel_002_100_10	100	10	516

Fonte: o autor, 2017.

Os tempos de processamento das tarefas nas máquinas dos casos da Tabela 6.2 (*Benchmark* de Heller) podem ser obtidos em *Operations Research Library* (OR Library).

Tabela 6.3 - Casos selecionados para testes: *Benchmark* de Reeves

Caso	Problema Teste	Número de Tarefas	Número de Máquinas	C_{max}
11	Rec_001_20_05	20	5	1247
12	Rec_003_20_05	20	5	1109
13	Rec_005_20_05	20	5	1242
14	Rec_007_20_10	20	10	1566
15	Rec_009_20_10	20	10	1537
16	Rec_011_20_10	20	10	1431
17	Rec_013_20_15	20	15	1930
18	Rec_015_20_15	20	15	1950
19	Rec_017_20_15	20	15	1902
20	Rec_019_30_10	30	10	2093
21	Rec_021_30_10	30	10	2017
22	Rec_023_30_10	30	10	2011
23	Rec_025_30_15	30	15	2513
24	Rec_027_30_15	30	15	2373
25	Rec_029_30_15	30	15	2287
26	Rec_031_50_10	50	10	3045
27	Rec_033_50_10	50	10	3114
28	Rec_035_50_10	50	10	3277
29	Rec_037_75_20	75	20	4951
30	Rec_039_75_20	75	20	5087
31	Rec_041_75_20	75	20	4960

Fonte: o autor, 2017.

Os tempos de processamento das tarefas nas máquinas dos casos da Tabela 6.3 (*Benchmark* de Reeves) podem ser obtidos em *Operations Research Library* (OR Library).

Tabela 6.4 – Casos selecionados para testes: *Benchmark* de Taillard (continua...)

Caso	Problema Teste	Número de Tarefas	Número de Máquinas	*Cmax*
32	Ta_001_20_05	20	5	*1278*
33	Ta_002_20_05	20	5	*1359*
34	Ta_003_20_05	20	5	*1081*
35	Ta_004_20_05	20	5	*1293*
36	Ta_005_20_05	20	5	*1235*
37	Ta_006_20_05	20	5	*1195*
38	Ta_007_20_05	20	5	*1234*
39	Ta_008_20_05	20	5	*1206*
40	Ta_009_20_05	20	5	*1230*
41	Ta_010_20_05	20	5	*1108*
42	Ta_011_20_10	20	10	*1582*
43	Ta_012_20_10	20	10	*1659*
44	Ta_013_20_10	20	10	*1496*
45	Ta_014_20_10	20	10	*1377*
46	Ta_015_20_10	20	10	*1419*
47	Ta_016_20_10	20	10	*1397*
48	Ta_017_20_10	20	10	*1484*
49	Ta_018_20_10	20	10	*1538*
50	Ta_019_20_10	20	10	*1593*
51	Ta_020_20_10	20	10	*1591*
52	Ta_021_20_20	20	20	*2297*
53	Ta_022_20_20	20	20	*2099*
54	Ta_023_20_20	20	20	*2326*
55	Ta_024_20_20	20	20	*2223*
56	Ta_025_20_20	20	20	*2291*
57	Ta_026_20_20	20	20	*2226*
58	Ta_027_20_20	20	20	*2273*
59	Ta_028_20_20	20	20	*2200*
60	Ta_029_20_20	20	20	*2237*
61	Ta_030_20_20	20	20	*2178*
62	Ta_031_50_05	50	5	*2724*
63	Ta_032_50_05	50	5	*2834*
64	Ta_033_50_05	50	5	*2621*
65	Ta_034_50_05	50	5	*2751*
66	Ta_035_50_05	50	5	*2863*
67	Ta_036_50_05	50	5	*2829*
68	Ta_037_50_05	50	5	*2725*
69	Ta_038_50_05	50	5	*2683*
70	Ta_039_50_05	50	5	*2552*
71	Ta_040_50_05	50	5	*2782*

Fonte: o autor, 2017.

Tabela 6.4 – Casos selecionados para testes: *Benchmark* de Taillard (fim...)

Caso	Problema Teste	Número de Tarefas	Número de Máquinas	C_{max}
72	Ta_041_50_10	50	10	2291
73	Ta_042_50_10	50	10	2867
74	Ta_043_50_10	50	10	2839
75	Ta_044_50_10	50	10	3063
76	Ta_045_50_10	50	10	2976
77	Ta_046_50_10	50	10	3006
78	Ta_047_50_10	50	10	3093
79	Ta_048_50_10	50	10	3037
80	Ta_049_50_10	50	10	2897
81	Ta_050_50_10	50	10	3065
82	Ta_051_50_20	50	20	3371
83	Ta_052_50_20	50	20	3368
84	Ta_053_50_20	50	20	3591
85	Ta_054_50_20	50	20	3635
86	Ta_055_50_20	50	20	3553
87	Ta_056_50_20	50	20	3667
88	Ta_057_50_20	50	20	3672
89	Ta_058_50_20	50	20	3627
90	Ta_059_50_20	50	20	3645
91	Ta_060_50_20	50	20	3696
92	Ta_061_100_05	100	5	5493
93	Ta_062_100_05	100	5	5268
94	Ta_063_100_05	100	5	5175
95	Ta_064_100_05	100	5	5014
96	Ta_065_100_05	100	5	5250
97	Ta_066_100_05	100	5	5135
98	Ta_067_100_05	100	5	5246
99	Ta_068_100_05	100	5	5094
100	Ta_069_100_05	100	5	5448
101	Ta_070_100_05	100	5	5322
102	Ta_071_100_10	100	10	5770
103	Ta_072_100_10	100	10	5349
104	Ta_073_100_10	100	10	5676
105	Ta_074_100_10	100	10	5781
106	Ta_075_100_10	100	10	5467
107	Ta_076_100_10	100	10	5303
108	Ta_077_100_10	100	10	5595
109	Ta_078_100_10	100	10	5617
110	Ta_079_100_10	100	10	5871

Fonte: o autor, 2017.

Os tempos de processamento das tarefas nas máquinas dos casos da Tabela 6.4 (*Benchmark* de Taillard) podem ser obtidos em *Operations Research Library* (OR Library).

6.2 PROCESSO DE GERAÇÃO DOS DADOS

Os algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 propostos nesta tese foram implementados e executados em ambiente computacional MATLAB® R2012b da MathWorks, no sistema operacional Windows 7 Home Premium de 64 bits instalado em um computador Acer® com um processador Intel Core i5 2450M, 2,5 GHz e 4GB de memória RAM (*Random Access Memory*).

Considerando que os algoritmos de Evolução Diferencial são métodos de otimização estocásticos, sendo, portanto, necessário que cada algoritmo seja executado várias vezes para uma mesma função com o intuito de reduzir o impacto da natureza estocástica dos algoritmos durante a avaliação de suas performances. Diante disto, os 110 casos (Tabelas 7.1 a 7.4) selecionados dos *Benchmarks* de Carlier (1978), Heller (1960), Reeves (1995) e Taillard (1993) foram executados 50 vezes cada.

Para todos os algoritmos, o tamanho da população, que constitui o conjunto de indivíduos cogitados como solução e usados para criar novo conjunto de indivíduos para análise, foi mantido em 50 membros (indivíduos), e o número de iterações (gerações) para cada experimento, que representa um contador de um ciclo completo de cada algoritmo, foi mantido em 2000.

Também foi utilizado o mesmo critério de parada em todos os experimentos, que no caso deste estudo foi o número máximo de avaliações da função objetivo, definido como D*10000, onde D indica a dimensão do problema, determinada pelo número de tarefas. Não foi estipulado um tempo limite para a execução dos algoritmos.

No algoritmo EDD-FSP a estratégia de mutação adotada foi a ED*/rand/1/bin* e os fatores de mutação (F) e de cruzamento (CR) foram configurados em 0,5 e 0,5, respectivamente. Ressalta-se que no algoritmo EDD-FSP os parâmetros F e Cr foram configurados tendo como base os resultados de um estudo preliminar (BOARETO; MORAIS; COELHO, 2017) que visou investigar a influência de

diferentes configurações dos parâmetros no desempenho do algoritmo de ED para minimização do *Makespan* em ambientes *Flow Shop* Permutacional.

No algoritmo EDDA-FSP$_1$ estratégia de mutação adotada foi a ED/*current-to-p-best*/*1/bin*, enquanto que no algoritmo EDDA-FSP$_2$ em que a cada geração uma estratégia de mutação é escolhida aleatoriamente com distribuição uniforme para cada vetor alvo, as estratégias adotadas foram a ED/*rand*/1/bin, a ED/*rand*/2/bin, a ED/*rand-to-best*/2/bin e a ED/*current-to-rand*/1/bin.

No algoritmo EDDA-FSP$_1$ os parâmetros μ_{CR} e μ_F, utilizados para gerar o fator de mutação F_i e a taxa de cruzamento CR_i associado a cada indivíduo, foram inicializados em 0,5. Ainda no algoritmo EDDA-FSP$_1$ o parâmetro *c* que controla a taxa de adaptação dos parâmetros foi configurado em 0,1 e o parâmetro *p* que determina a ganância da estratégia de mutação foi configurado em 0,05.

No algoritmo EDDA-FSP$_2$ o parâmetro *p*, que indica a probabilidade de uma estratégia de mutação ser escolhida na geração *G*, é inicializado em 0,25 e os parâmetros CRm_k utilizados para gerar as taxas de cruzamento CR_i, para cada vetor alvo na geração *G*, são iniciados em 0,5. O parâmetro *LP* que corresponde ao período de aprendizagem foi configurado em 50 gerações.

6.3 PROCESSO DE ANÁLISE DOS DADOS

O desempenho dos algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ propostos nesta tese foram avaliados em relação à qualidade das soluções produzidas, bem como em relação às informações de desvios padrão e a convergência dos algoritmos. Os algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ também foram avaliados em termos de tempos médios computacionais.

A qualidade das soluções fornecidas pelos algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ para o caso de minimização do *Makespan* em *Flow Shop* Permutacional foi avaliada em termos de erro percentual (E%) da função objetivo, ou seja, em termos da variação percentual entre a solução fornecida pelo método e a melhor solução (C_{max}) identificada na literatura especializada.

O erro relativo percentual (*E%)* da função objetivo foi obtido por meio da equação (7.1), obtida de Tien et al. (2015), tal que:

$$\text{Erro (E\%)} = \left(\frac{f(x^{pt}_{best}) - f(x^{*}_{UB})}{f(x^{*}_{UB})}\right).100 \qquad (7.1)$$

onde x^{pt}_{best} representa a melhor solução encontrada pelo algoritmo para um problema teste em todos os experimentos e x^{*}_{UB} representa a melhor solução (C_{max}) conhecida para o problema teste, que no caso deste estudo, são caracterizados pelos limitantes superiores conhecidos do problema.

Conforme anteriormente mencionado, outros indicativos utilizados para verificar o desempenho dos algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ propostos nesta tese foram os desvios padrão e a convergência. Os desvios padrão indicam a dispersão dos valores da função objetivo (C_{max}) em torno das médias, enquanto a convergência determina o momento (geração) em que os membros da população convergem para um valor comum.

O tempo médio computacional é calculado pela soma dos tempos de computação de cada problema dividida pelo número total de problemas resolvidos (média aritmética dos tempos de computação) (MORAIS; MOCCELLIN, 2010). Na experimentação computacional, o tempo médio de computação foi medido em segundos(s).

Capítulo 7 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados e discutidos os resultados produzidos pelos algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ propostos nesta tese. O desempenho dos algoritmos propostos é avaliado em termos de erros relativos percentuais, desvio padrão e convergência. Os tempos médios computacionais também foram considerados para avaliar o desempenho dos algoritmos propostos.

7.1 ERROS RELATIVOS PERCENTUAIS

Nas Tabelas 7.1 a 7.4 são apresentados as melhores soluções (x^{pt}_{best}) para os casos testes dos *Benchmarks* de Carlier, Heller, Reeves e Taillard produzidas pelos algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$, bem como os respectivos Erros Relativos Percentuais para cada caso teste.

Os Erros Relativos Percentuais Médios (ER% Médio) para os grupos de casos testes dos *Benchmarks* de Carlier, Heller, Reeves e Taillard são sumarizados na Tabela 7.5.

Tabela 7.1 - Erros Relativos Percentuais: Casos testes – *Benchmark* Carlier

Caso	Descrição	x^{*}_{UB} (C_{max})	EDD-FSP		EDDA-FSP$_1$		EDDA-FSP$_2$	
			x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*
1	Car_001_11_05	7038	7038	0,00	7038	0,00	7038	0,00
2	Car_002_13_04	7166	7166	0,00	7166	0,00	7166	0,00
3	Car_003_12_05	7312	7312	0,00	7312	0,00	7312	0,00
4	Car_004_14_04	8003	8003	0,00	8003	0,00	8003	0,00
5	Car_005_10_06	7720	7720	0,00	7720	0,00	7720	0,00
6	Car_006_08_09	8505	8505	0,00	8505	0,00	8505	0,00
7	Car_007_07_07	6590	6590	0,00	6590	0,00	6590	0,00
8	Car_008_08_08	8366	8366	0,00	8366	0,00	8366	0,00

Fonte: o autor, 2017.

Tabela 7.2 - Erros Relativos Percentuais: Casos testes – *Benchmark* Heller

Caso	Descrição	x^{*}_{UB} (C_{max})	EDD-FSP		EDDA-FSP$_1$		EDDA-FSP$_2$	
			x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*
9	Hel_001_20_10	136	136	0,00	135	-0,73	135	-0,73
10	Hel_002_100_10	516	518	0,38	515	-0,19	515	-0,19

Fonte: o autor, 2017.

Tabela 7.3 - Erros Relativos Percentuais: Casos testes – *Benchmark* Reeves

Caso	Descrição	x^*_{UB} (C_{max})	EDD-FSP		EDDA-FSP$_1$		EDDA-FSP$_2$	
			x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*
11	Rec_001_20_05	1247	1249	0,16	1247	0,00	1249	0,16
12	Rec_003_20_05	1109	1109	0,00	1109	0,00	1109	0,00
13	Rec_005_20_05	1242	1245	0,24	1245	0,24	1245	0,24
14	Rec_007_20_10	1566	1566	0,00	1566	0,00	1566	0,00
15	Rec_009_20_10	1537	1537	0,00	1537	0,00	1537	0,00
16	Rec_011_20_10	1431	1431	0,00	1431	0,00	1431	0,00
17	Rec_013_20_15	1930	1935	0,25	1934	0,20	1930	0,00
18	Rec_015_20_15	1950	1961	0,56	1950	0,00	1950	0,00
19	Rec_017_20_15	1902	1909	0,36	1909	0,36	1902	0,00
20	Rec_019_30_10	2093	2135	2,00	2101	0,38	2110	0,81
21	Rec_021_30_10	2017	2054	1,83	2040	1,14	2042	1,23
22	Rec_023_30_10	2011	2060	2,43	2021	0,49	2023	0,59
23	Rec_025_30_15	2513	2604	3,62	2542	1,15	2527	0,55
24	Rec_027_30_15	2373	2419	1,93	2396	0,96	2396	0,96
25	Rec_029_30_15	2287	2370	3,62	2298	0,48	2299	0,52
26	Rec_031_50_10	3045	3208	5,35	3114	2,26	3123	2,56
27	Rec_033_50_10	3114	3168	1,73	3125	0,35	3127	0,41
28	Rec_035_50_10	3277	3272	-0,15	3253	-0,73	3250	-0,82
29	Rec_037_75_20	4951	5470	10,48	5121	3,43	5248	5,99
30	Rec_039_75_20	5087	5444	7,01	5214	2,49	5250	3,20
31	Rec_041_75_20	4960	5447	9,81	5170	4,23	5232	5,48

Fonte: o autor, 2017.

Tabela 7.4 - Erros Relativos Percentuais: Casos testes – *Benchmark* Taillard (continua...)

Caso	Descrição	x^*_{UB} (C_{max})	EDD-FSP		EDDA-FSP$_1$		EDDA-FSP$_2$	
			x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*	x^{pt}_{best}	*E%*
32	Ta_001_20_05	1278	1278	0,00	1278	0,00	1582	23,78
33	Ta_002_20_05	1359	1359	0,00	1359	0,00	1659	22,07
34	Ta_003_20_05	1081	1081	0,00	1081	0,00	1496	38,39
35	Ta_004_20_05	1293	1293	0,00	1280	-1,00	1379	6,65
36	Ta_005_20_05	1235	1243	0,64	1235	0,00	1419	14,89
37	Ta_006_20_05	1195	1195	0,00	1210	1,25	1397	16,90
38	Ta_007_20_05	1234	1251	1,37	1239	0,40	1484	20,25
39	Ta_008_20_05	1206	1206	0,00	1206	0,00	1544	28,02
40	Ta_009_20_05	1230	1230	0,00	1230	0,00	1597	29,83
41	Ta_010_20_05	1108	1108	0,00	1108	0,00	1108	0,00
42	Ta_011_20_10	1582	1591	0,56	1582	0,00	1278	-19,21
43	Ta_012_20_10	1659	1660	0,06	1659	0,00	1359	-18,08
44	Ta_013_20_10	1496	1508	0,80	1496	0,00	1081	-27,74
45	Ta_014_20_10	1377	1385	0,58	1379	0,14	1280	-7,04
46	Ta_015_20_10	1419	1425	0,42	1419	0,00	1235	-12,96
47	Ta_016_20_10	1397	1406	0,64	1400	0,21	1195	-14,45
48	Ta_017_20_10	1484	1487	0,00	1484	-0,20	1239	-16,67
49	Ta_018_20_10	1538	1552	0,91	1544	0,39	1206	-21,58
50	Ta_019_20_10	1593	1608	0,94	1602	0,56	1230	-22,78
51	Ta_020_20_10	1591	1608	1,06	1604	0,81	1594	0,18
52	Ta_021_20_20	2297	2309	0,52	2298	0,04	2297	0,00
53	Ta_022_20_20	2099	2111	0,57	2100	0,04	2101	0,09
54	Ta_023_20_20	2326	2342	0,68	2331	0,21	2326	0,00
55	Ta_024_20_20	2223	2234	0,49	2223	0,00	2223	0,00

Fonte: o autor, 2017.

Tabela 7.4 - Erros Relativos Percentuais: Casos testes – *Benchmark* Taillard (fim...)

56	Ta_025_20_20	2291	2298	0,30	2294	0,13	2297	0,26
57	Ta_026_20_20	2226	2230	0,17	2229	0,13	2229	0,13
58	Ta_027_20_20	2273	2287	0,61	2277	0,17	2273	0,00
59	Ta_028_20_20	2200	2215	0,68	2204	0,18	2200	0,00
60	Ta_029_20_20	2237	2242	0,22	2237	0,00	2237	0,00
61	Ta_030_20_20	2178	2183	0,22	2178	0,00	2178	0,00
62	Ta_031_50_05	2724	2724	0,00	2724	0,00	2724	0,00
63	Ta_032_50_05	2834	2845	0,38	2838	0,14	2834	0,00
64	Ta_033_50_05	2621	2633	0,45	2621	0,00	2621	0,00
65	Ta_034_50_05	2751	2765	0,50	2751	0,00	2751	0,00
66	Ta_035_50_05	2863	2864	0,03	2863	0,00	2863	0,00
67	Ta_036_50_05	2829	2834	0,17	2829	0,00	2829	0,00
68	Ta_037_50_05	2725	2732	0,25	2725	0,00	2725	0,00
69	Ta_038_50_05	2683	2705	0,81	2683	0,00	2683	0,00
70	Ta_039_50_05	2552	2574	0,86	2554	0,07	2552	0,00
71	Ta_040_50_05	2782	2782	0,00	2782	0,00	2782	0,00
72	Ta_041_50_10	2291	3174	6,11	3053	2,07	3063	2,40
73	Ta_042_50_10	2867	3039	5,99	2932	2,26	2955	3,06
74	Ta_043_50_10	2839	3007	5,91	2904	2,28	2904	2,28
75	Ta_044_50_10	3063	3162	3,23	3082	0,62	3086	0,75
76	Ta_045_50_10	2976	3133	5,27	3024	1,61	3020	1,47
77	Ta_046_50_10	3006	3139	4,42	3064	1,92	3074	2,26
78	Ta_047_50_10	3093	3215	3,94	3140	1,51	3147	1,74
79	Ta_048_50_10	3037	3139	3,35	3058	0,69	3061	0,79
80	Ta_049_50_10	2897	3022	4,31	2943	1,58	2932	1,20
81	Ta_050_50_10	3065	3216	7,52	3131	4,68	3139	4,94
82	Ta_051_50_20	3371	4107	21,83	3951	17,20	3964	17,59
83	Ta_052_50_20	3368	3981	18,20	3821	13,45	3845	14,16
84	Ta_053_50_20	3591	3947	9,91	3769	4,95	3812	6,15
85	Ta_054_50_20	3635	3986	9,65	3826	5,25	3863	6,27
86	Ta_055_50_20	3553	3870	8,92	3728	4,92	3716	4,58
87	Ta_056_50_20	3667	3933	7,25	3784	3,19	3815	4,03
88	Ta_057_50_20	3672	3970	8,11	3807	3,67	3832	4,35
89	Ta_058_50_20	3627	3970	9,45	3809	5,01	3835	5,73
90	Ta_059_50_20	3645	3982	9,24	3837	5,26	3856	5,78
91	Ta_060_50_20	3696	4048	9,52	3856	4,32	3854	4,27
92	Ta_061_100_05	5493	5527	0,61	5493	0,00	5493	0,00
93	Ta_062_100_05	5268	5284	0,30	5274	0,11	5268	0,00
94	Ta_063_100_05	5175	5213	0,73	5175	0,00	5175	0,00
95	Ta_064_100_05	5014	5030	0,31	5018	0,07	5018	0,07
96	Ta_065_100_05	5250	5256	0,11	5250	0,00	5250	0,00
97	Ta_066_100_05	5135	5140	0,09	5135	0,00	5135	0,00
98	Ta_067_100_05	5246	5284	0,72	5246	0,00	5246	0,00
99	Ta_068_100_05	5094	5116	0,43	5094	0,00	5094	0,00
100	Ta_069_100_05	5448	5484	0,66	5448	0,00	5448	0,00
101	Ta_070_100_05	5322	5346	0,45	5325	0,05	5322	0,00
102	Ta_071_100_10	5770	5977	3,58	5791	0,36	5801	0,53
103	Ta_072_100_10	5349	5494	2,71	5320	-0,54	5318	-0,57
104	Ta_073_100_10	5676	5815	2,44	5679	0,05	5688	0,21
105	Ta_074_100_10	5781	6013	4,01	5860	1,36	5874	1,60
106	Ta_075_100_10	5467	5697	4,20	5520	0,96	5513	0,84
107	Ta_076_100_10	5303	5448	2,73	5310	0,13	5317	0,26
108	Ta_077_100_10	5595	5766	3,05	5623	0,50	5632	0,66
109	Ta_078_100_10	5617	5796	3,18	5683	1,17	5694	1,37
110	Ta_079_100_10	5871	5976	1,78	5922	0,86	5928	0,97

Fonte: o autor, 2017.

Tabela 7.5 - Erros Relativos Percentuais Médios (*E*% Médio) por *Benchmark*

Benchmark de casos testes	EDD-FSP	EDDA-FSP$_1$	EDDA-FSP$_2$
	E% Médio	*E%* Médio	*E%* Médio
Carlier	0,00	0,00	0,00
Heller	0,19	-0,46	-0,46
Reeves	2,44	0,83	0,70
Taillard	2,66	1,20	1,78

Fonte: o autor, 2017.

Para os casos testes do *Benchmark* de Carlier, conforme pode ser visualizado nas Tabelas 7.1 e 7.5, os algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FSP$_2$ atingiram as melhores soluções (C_{max}) conhecidas na literatura em 100% dos casos.

Conforme pode ser visualizado na Tabela 7.2, para os casos testes do *Benchmark* de Heller, o algoritmo EDD-FSP atingiu as melhores soluções (C_{max}) conhecidas na literatura para um caso (caso 9) e não foi capaz de alcançar as melhores soluções conhecidas na literatura para o outro caso (caso 10). Ainda em relação aos casos testes do *Benchmark* de Heller, os algoritmos EDDA-FSP$_1$ e EDDA-FSP$_2$ foram capazes de produzir soluções superiores as melhores soluções conhecidas na literatura para ambos os casos. Conforme pode ser visualizado na Tabela 7.5, os algoritmos EDDA-FSP$_1$ e EDDA-FSP$_2$ apresentaram desempenho idêntico e produziram erros relativos percentuais médios menores que o algoritmo EDD-FSP.

Para os casos do *Benchmark* de Reeves o algoritmo EDDA-FSP$_2$ forneceu os menores erros relativos percentuais médios que os algoritmos EDD-FSP e EDDA-FSP$_1$, conforme pode ser visualizado na Tabela 7.5. Na tabela 7.3 pode ser visto que para este grupo de casos testados, os algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FPS$_2$ atingiram as melhores soluções (C_{max}) conhecidas na literatura em 19,04%, 28,57% e 33,33% dos casos, respectivamente. Os algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FPS$_2$ produziram soluções superiores as melhores soluções (C_{max}) conhecidas na literatura em 4,76% dos casos.

Conforme apresentado na Tabela 7.5, para os casos do *Benchmark* de Taillard o algoritmo EDDA-FSP$_1$ obteve melhor desempenho, em termos de erros relativos percentuais médios, que os algoritmos EDD-FSP e EDDA-FSP$_2$. Para este grupo de casos testados, conforme pode ser visualizado na Tabela 7.4, os algoritmos EDD-FSP, EDDA-FSP$_1$ e EDDA-FPS$_2$ atingiram as melhores soluções

(C_{max}) conhecidas na literatura em 13,92%, 36,70% e 34,17% dos casos, respectivamente. Embora o algoritmo EDDA-FSP_1 tenha apresentado melhor desempenho em relação ao percentual de alcance das melhores soluções conhecidas, o algoritmo EDDA-FSP_2 superou as melhores soluções conhecidas na literatura em 12,65% dos casos, enquanto o EDDA-FSP_2 superou as melhores soluções em apenas 3,79% dos casos testados.

7.2 DESVIO PADRÃO

Outro indicativo utilizado para avaliar desempenho dos algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FPS_2 é o desvio padrão que pode ser observado nas Tabelas 7.6 a 7.9. Nestas tabelas também podem ser verificados, os melhores (Mínimo) e os piores (Máximo) valores produzidos pelos algoritmos, bem como as médias, para cada caso teste selecionado dos *Benchmarks* de Carlier, Heller, Reeves e Taillard. Os resultados apresentados nas Tabelas 7.6 a 7.9 foram obtidos após 50 replicações.

Tabela 7.6 – Desvio Padrão: Casos testes – *Benchmark* Carlier

Caso	Descrição	Algoritmo	Mínimo	Máximo	Média	Desvio Padrão
1	Car_001_11_05	EDD-FSP	7038	7038	7038	0,00
		EDDA-FSP_1	7038	7038	7038	0,00
		EDDA-FSP_2	7038	7038	7038	0,00
2	Car_002_13_04	EDD-FSP	7166	7166	7166	0,00
		EDDA-FSP_1	7166	7166	7166	0,00
		EDDA-FSP_2	7166	7166	7166	0,00
3	Car_003_12_05	EDD-FSP	7312	7312	7312	0,00
		EDDA-FSP_1	7312	7312	7312	0,00
		EDDA-FSP_2	7312	7312	7312	0,00
4	Car_004_14_04	EDD-FSP	8003	8003	8003	0,00
		EDDA-FSP_1	8003	8003	8003	0,00
		EDDA-FSP_2	8003	8003	8003	0,00
5	Car_005_10_06	EDD-FSP	7720	7821	7739,40	0,03
		EDDA-FSP_1	7720	7821	7732,12	31,33
		EDDA-FSP_2	7720	7767	7725,50	15,12
6	Car_006_08_09	EDD-FSP	8505	8570	8537,50	0,03
		EDDA-FSP_1	8505	8570	8518	26,26
		EDDA-FSP_2	8505	8570	8518	26,26
7	Car_007_07_07	EDD-FSP	6590	6590	6590	0,00
		EDDA-FSP_1	6590	6590	6590	0,00
		EDDA-FSP_2	6590	6590	6590	0,00
8	Car_008_08_08	EDD-FSP	8366	8366	8366	0,00
		EDDA-FSP_1	8366	8366	8366	0,00
		EDDA-FSP_2	8366	8366	8366	0,00

Fonte: o autor, 2017.

Tabela 7.7- Desvio Padrão: Casos testes – *Benchmark* Heller

Caso	Descrição	Algoritmo	Mínimo	Máximo	Média	Desvio Padrão
9	Hel_001_20_10	EDD-FSP	136	137	136,94	0,24
		EDDA-FSP$_1$	135	137	136,48	0,61
		EDDA-FSP$_2$	135	137	135,74	0,66
10	Hel_002_100_10	EDD-FSP	518	524	520,78	1,31
		EDDA-FSP$_1$	515	519	516,96	1,11
		EDDA-FSP$_2$	515	516	515,12	0,33

Fonte: o autor, 2017.

Tabela 7.8 – Desvio Padrão: Casos testes – *Benchmark* Reeves (continua...)

Caso	Descrição	Algoritmo	Mínimo	Máximo	Média	Desvio Padrão
11	Rec_001_20_05	EDD-FSP	1249	1249	1249	0,00
		EDDA-FSP$_1$	1247	1256	1249,14	1,05
		EDDA-FSP$_2$	1249	1249	1249	0,00
12	Rec_003_20_05	EDD-FSP	1109	1111	1109	0,00
		EDDA-FSP$_1$	1109	1111	1110,14	0,96
		EDDA-FSP$_2$	1109	1111	1109,6	0,91
13	Rec_005_20_05	EDD-FSP	1245	1245	1245	0,00
		EDDA-FSP$_1$	1245	1245	1245	0,00
		EDDA-FSP$_2$	1245	1245	1245	0,00
14	Rec_007_20_10	EDD-FSP	1566	1631	1580,7	0,01
		EDDA-FSP$_1$	1566	1584	1576,44	8,20
		EDDA-FSP$_2$	1566	1584	1567,9	5,44
15	Rec_009_20_10	EDD-FSP	1537	1547	1538,8	0,00
		EDDA-FSP$_1$	1537	1574	1541,5	11,21
		EDDA-FSP$_2$	1537	1539	1537,1	0,34
16	Rec_011_20_10	EDD-FSP	1431	1469	1438,6	0,01
		EDDA-FSP$_1$	1431	1454	1433,7	5,68
		EDDA-FSP$_2$	1431	1431	1431	0,00
17	Rec_013_20_15	EDD-FSP	1935	1969	1947	0,01
		EDDA-FSP$_1$	1934	1972	1942,68	7,79
		EDDA-FSP$_2$	1930	1942	1935,3	1,97
18	Rec_015_20_15	EDD-FSP	1961	2005	1971,9	0,01
		EDDA-FSP$_1$	1950	1988	1966,7	8,53
		EDDA-FSP$_2$	1950	1973	1958,7	6,11
19	Rec_017_20_15	EDD-FSP	1909	1958	1940,4	0,01
		EDDA-FSP$_1$	1909	1943	1924,46	10,43
		EDDA-FSP$_2$	1902	1939	1920,4	10,13
20	Rec_019_30_10	EDD-FSP	2135	2183	2163,2	0,01
		EDDA-FSP$_1$	2101	2126	2117,32	5,84
		EDDA-FSP$_2$	2110	2136	2120	5,34
21	Rec_021_30_10	EDD-FSP	2054	2078	2066,6	0,01
		EDDA-FSP$_1$	2040	2050	2048,72	2,12
		EDDA-FSP$_2$	2042	2050	2049,1	1,82
22	Rec_023_30_10	EDD-FSP	2060	2086	2070,9	0,00
		EDDA-FSP$_1$	2021	2053	2033,68	7,81
		EDDA-FSP$_2$	2023	2055	2038,6	9,56

Fonte: o autor, 2017.

Tabela 7.8 – Desvio Padrão: Casos testes – *Benchmark* Reeves (fim...)

		EDD-FSP	2604	2654	2634,5	0,01
		EDDA-FSP$_1$	2542	2577	2558,96	8,86
23	Rec_025_30_15	EDDA-FSP$_2$	2527	2594	2564,1	15,63
		EDD-FSP	2419	2463	2440,6	0,01
		EDDA-FSP$_1$	2396	2421	2407,06	6,21
24	Rec_027_30_15	EDDA-FSP$_2$	2396	2426	2404,7	6,33
		EDD-FSP	2370	2421	2393,7	0,01
		EDDA-FSP$_1$	2298	2396	2327,18	16,31
25	Rec_029_30_15	EDDA-FSP$_2$	2299	2349	2318,7	11,65
		EDD-FSP	3208	3264	3239,7	0,001
		EDDA-FSP$_1$	3114	3141	3125,18	5,84
26	Rec_031_50_10	EDDA-FSP$_2$	3123	3171	3142,9	10,32
		EDD-FSP	3168	3215	3190,6	0,01
		EDDA-FSP$_1$	3125	3172	3140,06	6,58
27	Rec_033_50_10	EDDA-FSP$_2$	3127	3147	3139,3	3,82
		EDD-FSP	3272	3299	3284,6	0,008
		EDDA-FSP$_1$	3253	3277	3265,94	4,04
28	Rec_035_50_10	EDDA-FSP$_2$	3250	3267	3261,3	4,39
		EDD-FSP	5470	5557	5508,4	0,02
		EDDA-FSP$_1$	5121	5233	5192,32	20,26
29	Rec_037_75_20	EDDA-FSP$_2$	5248	5352	5294,2	27,90
		EDD-FSP	5444	5521	5491,1	0,017
		EDDA-FSP$_1$	5214	5305	5264,9	20,87
30	Rec_039_75_20	EDDA-FSP$_2$	5250	5405	5340,4	31,52
		EDD-FSP	5447	5520	5487	0,017
		EDDA-FSP$_1$	5170	5241	5206,06	17,48
31	Rec_041_75_20	EDDA-FSP$_2$	5232	5343	5290	26,09

Fonte: o autor, 2017.

Tabela 7.9 - Desvio Padrão: Casos testes – *Benchmark* Taillard (continua...)

Caso	Descrição	Algoritmo	Mínimo	Máximo	Média	Desvio Padrão
		EDD-FSP	1278	1297	1299,6	0,002
		EDDA-FSP$_1$	1278	1297	1281,83	7,71
32	Ta_001_20_05	EDDA-FSP$_2$	1582	1597	1587,5	4,15
		EDD-FSP	1359	1366	1363,1	0,003
		EDDA-FSP$_1$	1359	1366	1361,53	3,26
33	Ta_002_20_05	EDDA-FSP$_2$	1659	1679	1668,2	5,55
		EDD-FSP	1081	1100	1087,4	0,005
		EDDA-FSP$_1$	1081	1098	1091,96	5,25
34	Ta_003_20_05	EDDA-FSP$_2$	1496	1518	1507,8	4,26
		EDD-FSP	1293	1302	1298,6	0,004
		EDDA-FSP$_1$	1280	1302	1295,43	5,30
35	Ta_004_20_05	EDDA-FSP$_2$	1379	1392	1384,3	3,96
		EDD-FSP	1243	1250	1249,7	0,001
		EDDA-FSP$_1$	1235	1243	1241,66	3,03
36	Ta_005_20_05	EDDA-FSP$_2$	1419	1434	1426,6	4,35

Fonte: o autor, 2017.

Tabela 7.9 – Desvio Padrão: Casos testes – *Benchmark* Taillard (continua...)

		EDD-FSP	1195	1210	1209,5	0,002
		EDDA-FSP$_1$	1210	1210	1210	0,00
37	Ta_006_20_05	EDDA-FSP$_2$	1397	1412	1402,4	3,80
		EDD-FSP	1251	1259	1251,8	0,002
		EDDA-FSP$_1$	1239	1251	1250,6	2,19
38	Ta_007_20_05	EDDA-FSP$_2$	1484	1499	1485,8	2,41
		EDD-FSP	1206	1211	1208,7	0,002
		EDDA-FSP$_1$	1206	1227	1211	4,01
39	Ta_008_20_05	EDDA-FSP$_2$	1544	1561	1551,1	3,98
		EDD-FSP	1230	1254	1248	0,009
		EDDA-FSP$_1$	1230	1261	1240,56	11,22
40	Ta_009_20_05	EDDA-FSP$_2$	1597	1615	1609	4,40
		EDD-FSP	1108	1131	1115,7	0,007
		EDDA-FSP$_1$	1108	1127	1111,66	6,21
41	Ta_010_20_05	EDDA-FSP$_2$	1108	1108	1108	29,98
		EDD-FSP	1591	1618	1603,2	0,008
		EDDA-FSP$_1$	1582	1618	1594,06	12,01
42	Ta_011_20_10	EDDA-FSP$_2$	1278	1297	1279,9	5,75
		EDD-FSP	1660	1691	1675,7	0,007
		EDDA-FSP$_1$	1659	1692	1673,3	9,67
43	Ta_012_20_10	EDDA-FSP$_2$	1359	1366	1359,3	1,03
		EDD-FSP	1508	1542	1517,9	0,006
		EDDA-FSP$_1$	1496	1540	1511,66	8,42
44	Ta_013_20_10	EDDA-FSP$_2$	1081	1088	1086,9	2,59
		EDD-FSP	1385	1415	1397,6	0,007
		EDDA-FSP$_1$	1379	1399	1389,56	5,56
45	Ta_014_20_10	EDDA-FSP$_2$	1280	1300	1292,4	3,55
		EDD-FSP	1425	1451	1435,1	0,005
		EDDA-FSP$_1$	1419	1448	1430,5	7,64
46	Ta_015_20_10	EDDA-FSP$_2$	1235	1244	1239,4	4,01
		EDD-FSP	1406	1424	1417,1	0,006
		EDDA-FSP$_1$	1400	1424	1407,7	6,99
47	Ta_016_20_10	EDDA-FSP$_2$	1195	1210	1208,5	4,54
		EDD-FSP	1487	1526	1494,5	0,006
		EDDA-FSP$_1$	1484	1504	1490,33	6,26
48	Ta_017_20_10	EDDA-FSP$_2$	1239	1251	1250,3	2,57
		EDD-FSP	1552	1586	1565,8	0,006
		EDDA-FSP$_1$	1544	1559	1552,46	4,11
49	Ta_018_20_10	EDDA-FSP$_2$	1206	1215	1207,1	2,24
		EDD-FSP	1608	1624	1612,7	0,003
		EDDA-FSP$_1$	1602	1636	1614,16	7,87
50	Ta_019_20_10	EDDA-FSP$_2$	1230	1236	1230,2	1,18
		EDD-FSP	1608	1623	1615,5	0,003
		EDDA-FSP$_1$	1604	1628	1611	6,19
51	Ta_020_20_10	EDDA-FSP$_2$	1594	1614	1607,3	4,59
		EDD-FSP	2309	2345	2322,1	0,006
		EDDA-FSP$_1$	2298	2341	2313,83	9,72
52	Ta_021_20_20	EDDA-FSP$_2$	2297	2322	2310	4,43
		EDD-FSP	2111	2141	2124,7	0,007
		EDDA-FSP$_1$	2100	2144	2113,7	8,90
53	Ta_022_20_20	EDDA-FSP$_2$	2101	2119	2111,6	5,01
		EDD-FSP	2342	2393	2371,8	0,01
		EDDA-FSP$_1$	2331	2366	2345,93	8,84
54	Ta_023_20_20	EDDA-FSP$_2$	2326	2359	2346,5	7,12

Fonte: o autor, 2017.

Tabela 7.9 – Desvio Padrão: Casos testes – *Benchmark* Taillard (continua...)

		EDD-FSP	2234	2269	2250,7	0,007
		EDDA-FSP$_1$	2223	2255	2238,1	6,60
55	Ta_024_20_20	EDDA-FSP$_2$	2223	2243	2233	4,26
		EDD-FSP	2298	2339	2318,4	0,008
		EDDA-FSP$_1$	2294	2316	2304,66	6,57
56	Ta_025_20_20	EDDA-FSP$_2$	2297	2313	2304,7	4,80
		EDD-FSP	2230	2248	2238	0,005
		EDDA-FSP$_1$	2229	2270	2239,76	7,89
57	Ta_026_20_20	EDDA-FSP$_2$	2229	2246	2236,5	3,79
		EDD-FSP	2287	2337	2309	0,009
		EDDA-FSP$_1$	2277	2327	2292,53	9,89
58	Ta_027_20_20	EDDA-FSP$_2$	2273	2303	2290,9	6,98
		EDD-FSP	2215	2258	2236	0,008
		EDDA-FSP$_1$	2204	2229	2216,13	6,32
59	Ta_028_20_20	EDDA-FSP$_2$	2200	2224	2213,4	6,14
		EDD-FSP	2242	2263	2246,5	0,004
		EDDA-FSP$_1$	2237	2284	2247,2	11,50
60	Ta_029_20_20	EDDA-FSP$_2$	2237	2245	2241,7	1,22
		EDD-FSP	2183	2230	2207,3	0,01
		EDDA-FSP$_1$	2178	2209	2189	7,83
61	Ta_030_20_20	EDDA-FSP$_2$	2178	2200	2186,4	5,43
		EDD-FSP	2724	2745	2735,6	0,003
		EDDA-FSP$_1$	2724	2735	2725,53	2,78
62	Ta_031_50_05	EDDA-FSP$_2$	2724	2729	2724,1	0,70
		EDD-FSP	2845	2882	2854,6	0,001
		EDDA-FSP$_1$	2838	2863	2844,43	6,85
63	Ta_032_50_05	EDDA-FSP$_2$	2834	2838	2837,4	1,10
		EDD-FSP	2633	2664	2645,9	0,006
		EDDA-FSP$_1$	2621	2641	2623,73	4,87
64	Ta_033_50_05	EDDA-FSP$_2$	2621	2622	2621,2	0,43
		EDD-FSP	2765	2782	2772,8	0,005
		EDDA-FSP$_1$	2751	2777	2760,16	5,80
65	Ta_034_50_05	EDDA-FSP$_2$	2751	2762	2757	4,82
		EDD-FSP	2864	2922	2885,5	0,02
		EDDA-FSP$_1$	2863	2864	2863,7	0,46
66	Ta_035_50_05	EDDA-FSP$_2$	2863	2864	2863,6	0,49
		EDD-FSP	2834	2896	2848,9	0,01
		EDDA-FSP$_1$	2829	2836	2831,76	1,94
67	Ta_036_50_05	EDDA-FSP$_2$	2829	2835	2830	1,47
		EDD-FSP	2732	2767	2741,7	0,009
		EDDA-FSP$_1$	2725	2745	2729,26	4,54
68	Ta_037_50_05	EDDA-FSP$_2$	2725	2725	2725	0,00
		EDD-FSP	2705	2730	2714	0,006
		EDDA-FSP$_1$	2683	2706	2699,5	7,30
69	Ta_038_50_05	EDDA-FSP$_2$	2683	2705	2697,2	7,99
		EDD-FSP	2574	2607	2588,6	0,008
		EDDA-FSP$_1$	2554	2566	2560,7	3,74
70	Ta_039_50_05	EDDA-FSP$_2$	2552	2564	2559,4	2,96
		EDD-FSP	2782	2782	2782	0,00
		EDDA-FSP$_1$	2782	2802	2783,66	5,14
71	Ta_040_50_05	EDDA-FSP$_2$	2782	2782	2782	0,00

Fonte: o autor, 2017.

Tabela 7.9 – Desvio Padrão: Casos testes – *Benchmark* Taillard (continua...)

		EDD-FSP	3174	3219	3200,9	0,009
		EDDA-FSP$_1$	3053	3107	3073,53	12,62
72	Ta_041_50_10	EDDA-FSP$_2$	3063	3142	3108,9	15,95
		EDD-FSP	3039	3088	3066,1	0,01
		EDDA-FSP$_1$	2932	2975	2958,43	11,47
73	Ta_042_50_10	EDDA-FSP$_2$	2955	3008	2979,7	10,07
		EDD-FSP	3007	3072	3041,8	0,01
		EDDA-FSP$_1$	2904	2953	2923,86	14,12
74	Ta_043_50_10	EDDA-FSP$_2$	2904	2978	2950,9	15,81
		EDD-FSP	3162	3205	3186,2	0,01
		EDDA-FSP$_1$	3082	3113	3101,93	9,05
75	Ta_044_50_10	EDDA-FSP$_2$	3086	3146	3113,6	9,27
		EDD-FSP	3133	3180	3157,2	0,01
		EDDA-FSP$_1$	3024	3076	3047,9	12,93
76	Ta_045_50_10	EDDA-FSP$_2$	3020	3110	3065,7	17,52
		EDD-FSP	3139	3191	3163,6	0,01
		EDDA-FSP$_1$	3064	3095	3081,23	8,93
77	Ta_046_50_10	EDDA-FSP$_2$	3074	3112	3093,1	10,21
		EDD-FSP	3215	3274	3250,4	0,01
		EDDA-FSP$_1$	3140	3176	3163,3	7,90
78	Ta_047_50_10	EDDA-FSP$_2$	3147	3202	3168,1	9,02
		EDD-FSP	3139	3196	3176,2	0,01
		EDDA-FSP$_1$	3058	3088	3072,63	7,01
79	Ta_048_50_10	EDDA-FSP$_2$	3061	3118	3086,1	12,47
		EDD-FSP	3022	3078	3047,9	0,01
		EDDA-FSP$_1$	2943	2995	2965,5	11,25
80	Ta_049_50_10	EDDA-FSP$_2$	2932	2989	2966,1	13,21
		EDD-FSP	3216	3255	3240,6	0,006
		EDDA-FSP$_1$	3131	3166	3145,6	8,55
81	Ta_050_50_10	EDDA-FSP$_2$	3139	3189	3163,5	13,19
		EDD-FSP	4107	4159	4136,6	0,01
		EDDA-FSP$_1$	3951	3996	3975,86	12,18
82	Ta_051_50_20	EDDA-FSP$_2$	3964	4035	4009,1	15,79
		EDD-FSP	3981	4048	4016,7	0,01
		EDDA-FSP$_1$	3821	3881	3851,96	14,26
83	Ta_052_50_20	EDDA-FSP$_2$	3845	3948	3898,4	21,85
		EDD-FSP	3947	3996	3975,2	0,01
		EDDA-FSP$_1$	3769	3822	3792,5	11,90
84	Ta_053_50_20	EDDA-FSP$_2$	3812	3869	3840,8	13,27
		EDD-FSP	3986	4053	4026,9	0,01
		EDDA-FSP$_1$	3826	3876	3846,9	12,71
85	Ta_054_50_20	EDDA-FSP$_2$	3863	3932	3899,2	16,59
		EDD-FSP	3870	3962	3922,1	0,01
		EDDA-FSP$_1$	3728	3788	3751,66	15,46
86	Ta_055_50_20	EDDA-FSP$_2$	3716	3837	3795,3	27,002
		EDD-FSP	3933	3993	3965,6	0,01
		EDDA-FSP$_1$	3784	3845	3807,83	14,14
87	Ta_056_50_20	EDDA-FSP$_2$	3815	3885	3850,4	16,2
		EDD-FSP	3970	4024	4001,3	0,01
88		EDDA-FSP$_1$	3807	3861	3841,63	11,83
	Ta_057_50_20	EDDA-FSP$_2$	3832	3917	3884,6	19,34

Fonte: o autor, 2017.

Tabela 7.9 – Desvio Padrão: Casos testes – *Benchmark* Taillard (continua...)

		EDD-FSP	3970	4048	4015,6	0,01
		EDDA-FSP$_1$	3809	3857	3836,03	11,70
89	Ta_058_50_20	EDDA-FSP$_2$	3835	3927	3886,9	19,59
		EDD-FSP	3982	4086	4043,4	0,0197
		EDDA-FSP$_1$	3837	3925	3877,9	16,49
90	Ta_059_50_20	EDDA-FSP$_2$	3856	3967	3931,6	22,14
		EDD-FSP	4048	4104	4072,2	0,01
		EDDA-FSP$_1$	3856	3911	3887,83	14,08
91	Ta_060_50_20	EDDA-FSP$_2$	3854	3950	3907,5	18,34
		EDD-FSP	5527	5559	5528,9	0,07
		EDDA-FSP$_1$	5493	5495	5493,46	0,86
92	Ta_061_100_05	EDDA-FSP$_2$	5493	5493	5493	0,00
		EDD-FSP	5284	5316	5299,1	0,01
		EDDA-FSP$_1$	5274	5289	5281,2	4,83
93	Ta_062_100_05	EDDA-FSP$_2$	5268	5289	5278,9	4,56
		EDD-FSP	5213	5221	5220,6	0,001
		EDDA-FSP$_1$	5175	5200	5181,7	7,66
94	Ta_063_100_05	EDDA-FSP$_2$	5175	5193	5176,6	4,009
		EDD-FSP	5030	5060	5053,5	0,009
		EDDA-FSP$_1$	5018	5035	5021,26	3,38
95	Ta_064_100_05	EDDA-FSP$_2$	5018	5023	5018,3	1,13
		EDD-FSP	5256	5284	5269,5	0,007
		EDDA-FSP$_1$	5250	5255	5252,56	2,28
96	Ta_065_100_05	EDDA-FSP$_2$	5250	5253	5252,1	1,26
		EDD-FSP	5140	5174	5148,1	0,006
		EDDA-FSP$_1$	5135	5146	5137,96	3,67
97	Ta_066_100_05	EDDA-FSP$_2$	5135	5139	5135,8	1,61
		EDD-FSP	5284	5332	5323,5	0,009
		EDDA-FSP$_1$	5246	5284	5254	9,53
98	Ta_067_100_05	EDDA-FSP$_2$	5246	5261	5252,3	6,22
		EDD-FSP	5116	5144	5138,2	0,007
		EDDA-FSP$_1$	5094	5128	5101,23	6,39
99	Ta_068_100_05	EDDA-FSP$_2$	5094	5106	5098,8	3,92
		EDD-FSP	5484	5514	5492,9	0,005
		EDDA-FSP$_1$	5448	5467	5454,3	6,44
100	Ta_069_100_05	EDDA-FSP$_2$	5448	5454	5451,6	2,87
		EDD-FSP	5346	5403	5357,5	0,01
		EDDA-FSP$_1$	5325	5342	5332,03	4,2
101	Ta_070_100_05	EDDA-FSP$_2$	5322	5334	5327,5	2,28
		EDD-FSP	5977	6049	6010	0,01
		EDDA-FSP$_1$	5791	5848	5825,96	14,38
102	Ta_071_100_10	EDDA-FSP$_2$	5801	5860	5833,6	16,9
		EDD-FSP	5511	5575	5538,6	0,01
		EDDA-FSP$_1$	5320	5364	5339,86	9,46
103	Ta_072_100_10	EDDA-FSP$_2$	5318	5382	5344,9	12,81
		EDD-FSP	5815	5856	5835,4	0,008
		EDDA-FSP$_1$	5679	5739	5705,63	14,07
104	Ta_073_100_10	EDDA-FSP$_2$	5688	5745	5707,3	18,54
		EDD-FSP	6013	6080	6047,5	0,01
		EDDA-FSP$_1$	5860	5926	5891,9	16,48
105	Ta_074_100_10	EDDA-FSP$_2$	5874	5951	5910,4	18,7

Fonte: o autor, 2017.

Tabela 7.9 – Desvio Padrão: Casos testes – *Benchmark* Taillard (...fim)

		EDD-FSP	5697	5792	5759,6	0,01
		EDDA-FSP_1	5520	5587	5551,26	18,53
106	Ta_075_100_10	EDDA-FSP_2	5513	5606	5557,1	22,15
		EDD-FSP	5448	5534	5491,4	0,01
		EDDA-FSP_1	5310	5344	5328,46	6,35
107	Ta_076_100_10	EDDA-FSP_2	5317	5372	5335,3	11,75
		EDD-FSP	5766	5812	5787,7	0,01
		EDDA-FSP_1	5623	5678	5655,23	11,95
108	Ta_077_100_10	EDDA-FSP_2	5632	5692	5666,1	15,72
		EDD-FSP	5796	5884	5842,1	0,01
		EDDA-FSP_1	5683	5716	5696,63	7,09
109	Ta_078_100_10	EDDA-FSP_2	5694	5739	5711,5	14,32
		EDD-FSP	5976	6019	6002,9	0,009
		EDDA-FSP_1	5922	5962	5939,26	12,3
110	Ta_079_100_10	EDDA-FSP_2	5928	5973	5951,5	10,49

Fonte: o autor, 2017.

Em relação aos desvios padrão, o algoritmo EDDA-FSP_2 apresentou melhor desempenho para os casos testes dos *Benchmarks* de Carlier e de Heller.

Para os casos testes do *Benchmark* de Carlier, em 6 dos 8 casos testes selecionados do *Benchmark* de Carlier todos os algoritmos propostos apresentam desvio padrão igual a 0,00. Para este grupo de casos testados, o algoritmo EDDA-FSP_1 apresentou os maiores desvios padrão, com valor de 31,33 para o caso 5 e de 26,26 para o caso 6.

Para os casos testes do *Benchmark* de Heller o algoritmo EDDA-FSP_1 apresentou o maior desvio padrão, com valor de 1,11 para o caso 10.

O algoritmo EDDA-FSP_1 apresentou o melhor desempenho para os casos testes do *Benchmark* de Reeves. O algoritmo EDDA-FSP_2 apresentou o maior desvio padrão, com valor de 31,52 para o caso 30, de 27,90 para o caso 29 e de 26,09 para o caso 31. Todos os algoritmos propostos apresentaram desvio padrão igual a 0,00 para o caso 13.

Para os casos testes do *Benchmark* de Taillard, o algoritmo EDDA-FSP_1 apresentou o melhor desempenho, pois os maiores valores para os desvios padrão foram produzidos pelo algoritmo EDDA-FSP_2, conforme pode ser visto na Tabela 7,9. Dentre os casos para os quais o algoritmo EDDA-FSP_2 apresenta os maiores valores para os desvios padrão destaca-se os seguintes: caso 41 – desvio padrão 29,98; caso 86 – desvio padrão 27,002; caso 106 – desvio padrão 22,15; caso 90 – desvio padrão 22,14; caso 83 – desvio padrão 21,85; caso 89 – desvio padrão 19,59; caso 104 – desvio padrão 18,54; e caso 91 – desvio padrão 18,34.

7.3 CONVERGÊNCIA

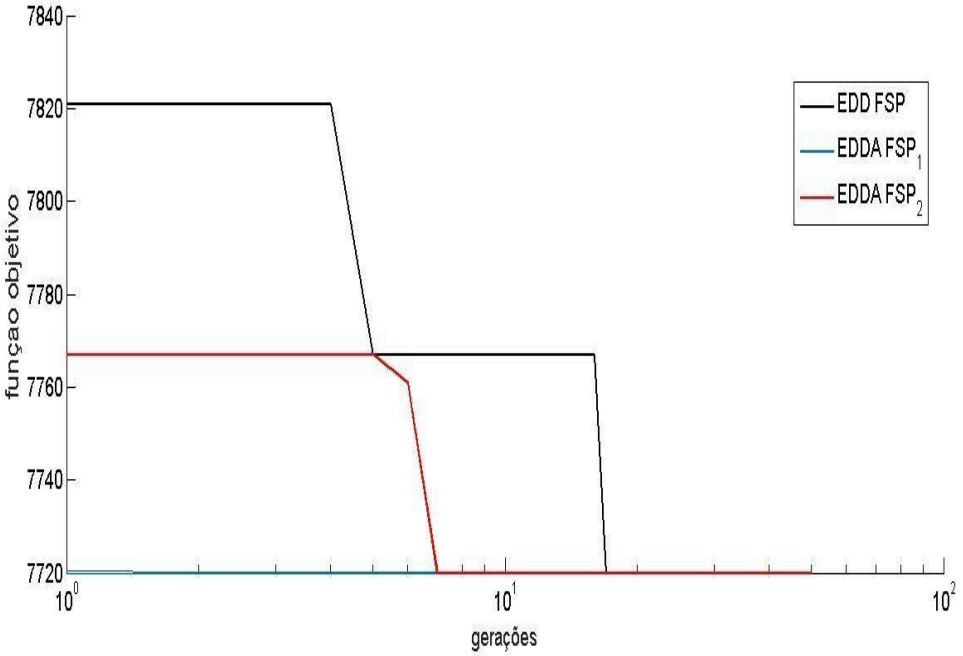
Uma análise da convergência dos algoritmos propostos constitui outra forma de se observar o desempenho dos algoritmos propostos.

Para a análise da convergência dos algoritmos, dos 110 casos testados foram selecionados 10, conforme segue: Caso 5 - Car_005_10_06; Caso 9 - Hel_001_20_10; Caso 10 - Hel_002_100_10; Caso 21 - Rec_021_30_10; Caso 31 - Rec_041_75_20; Caso 36 - Ta_005_20_05; Caso 41 - Ta_010_20_05; Caso 55 - Ta_024_20_20; Caso 94 - Ta_063_100_05; e Caso 110 - Ta_079_100_10.

As curvas de convergência para as melhores soluções fornecidas pelos algoritmos propostos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 podem ser visualizadas nas figuras 7.1 a 7.10.

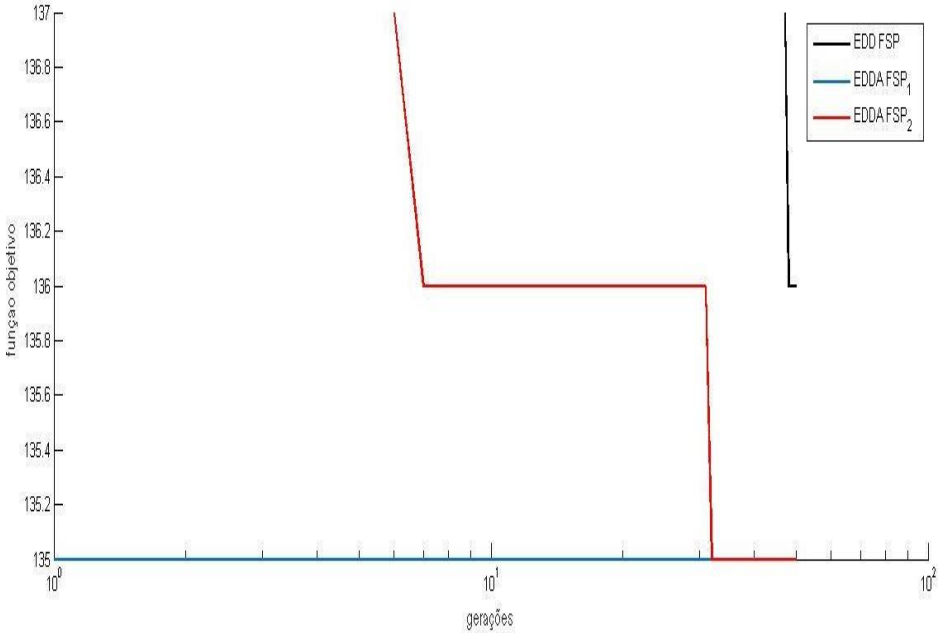
Nas Figuras 7.1 a 7.10 o eixo das abscissas representa o número de iterações que o algoritmo obteve em escala logarítmica e o eixo das ordenadas representa o valor médio da função objetivo, sendo minimizada durante o processo de otimização. A escala logarítmica, adotada no eixo das abscissas, justifica-se pelo fato de que o logaritmo reduz a representação de escala linear possibilitando assim que as linhas de convergência dos algoritmos sejam melhor visualizadas.

Figura 7.1 – Convergência: Caso 5 - Car_005_10_06



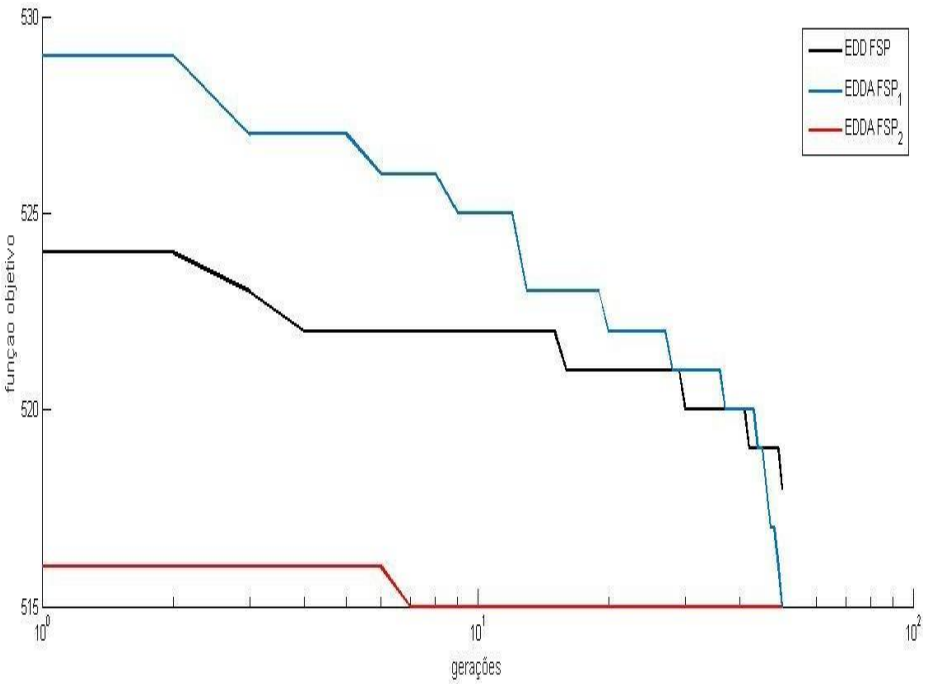
Fonte: o autor, 2017.

Figura 7.2 – Convergência: Caso 9 - Hel_001_20_10



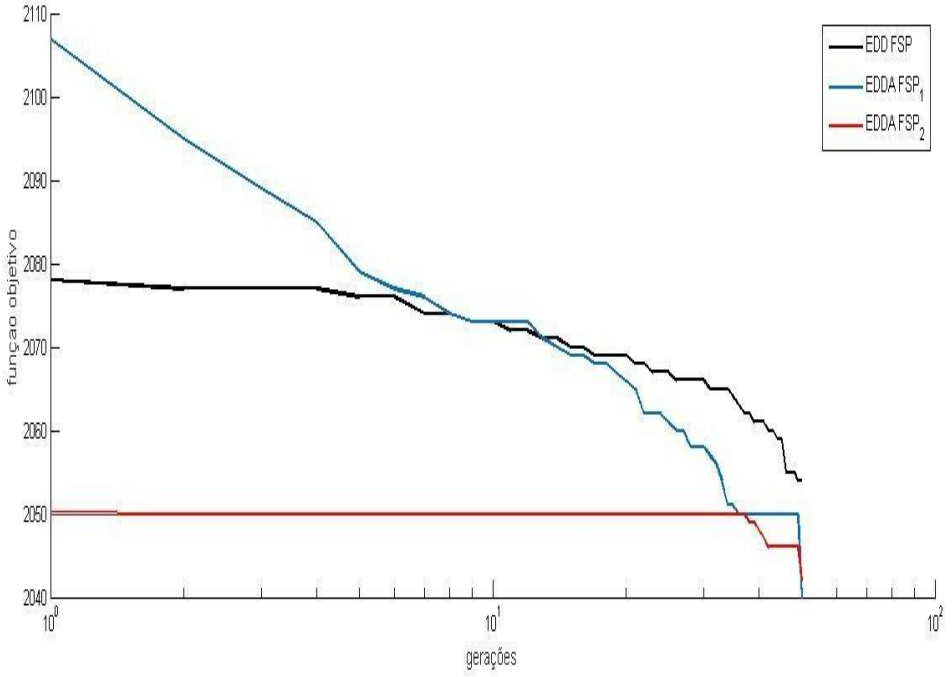
Fonte: o autor, 2017.

Figura 7.3 – Convergência: Caso 10 - Hel_002_100_10



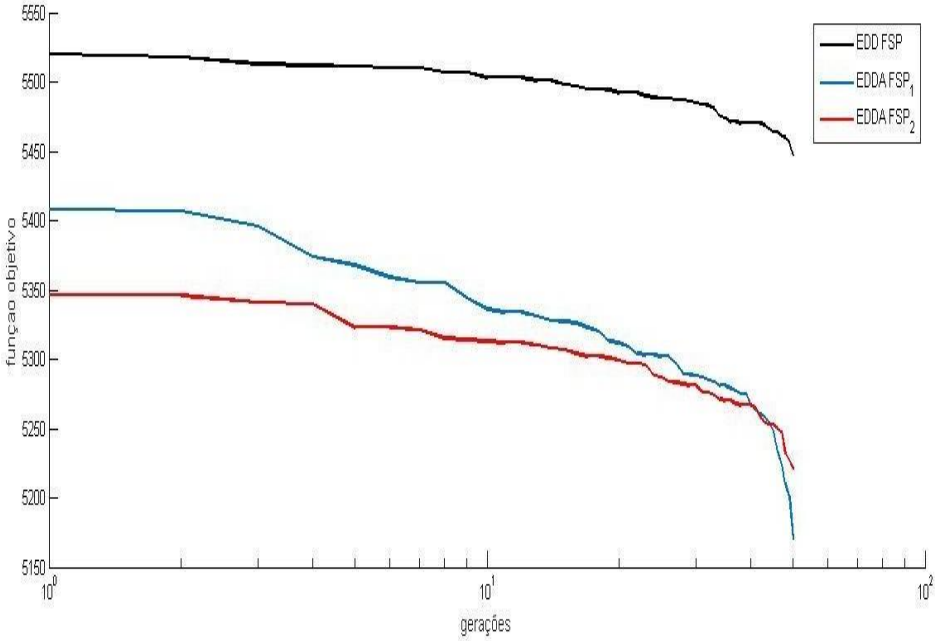
Fonte: o autor, 2017.

Figura 7.4 – Convergência: Caso 21 - Rec_021_30_10



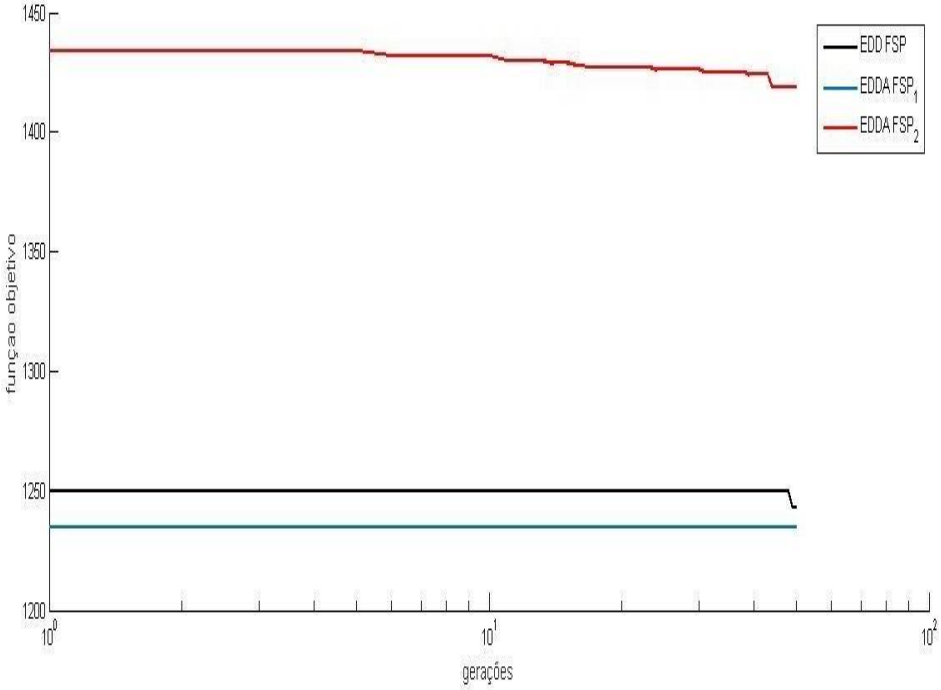
Fonte: o autor, 2017.

Figura 7.5 – Convergência: Caso 31 - Rec_041_75_20



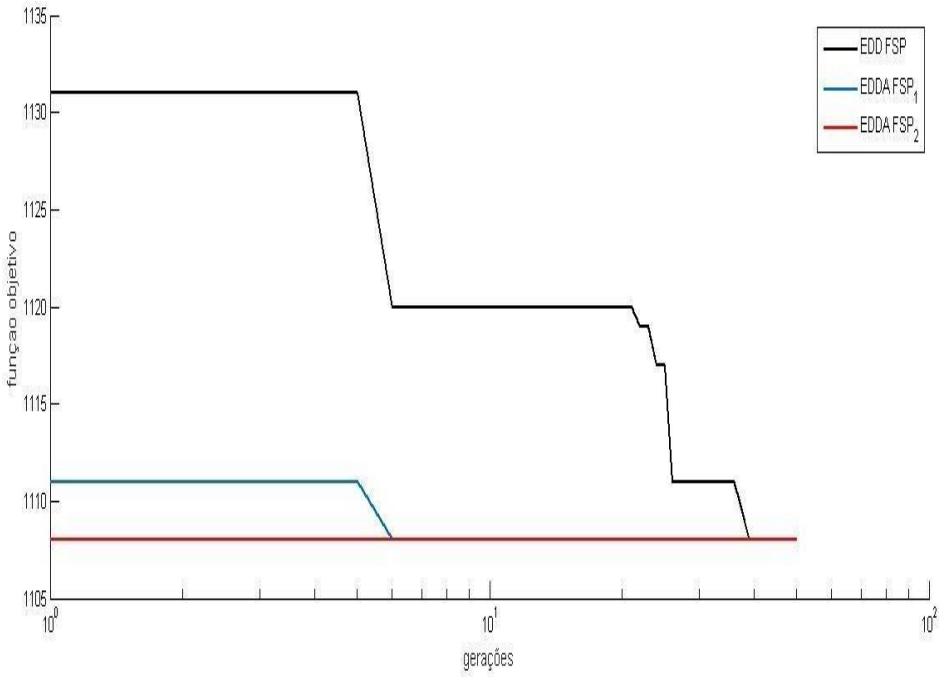
Fonte: o autor, 2017.

Figura 7.6 – Convergência: Caso 36 - Ta_005_20_05



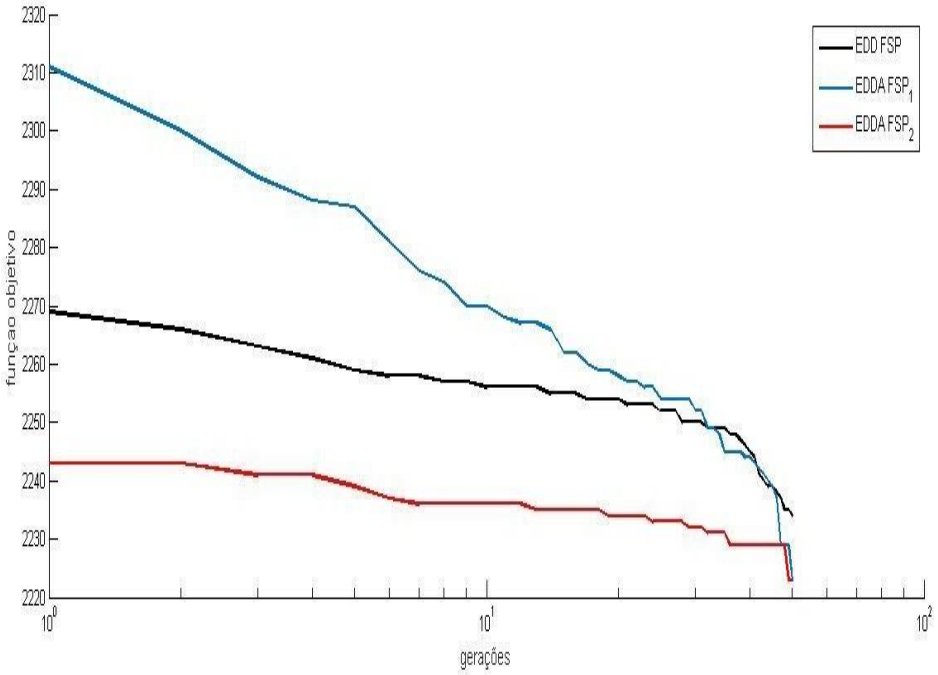
Fonte: o autor, 2017.

Figura 7.7 – Convergência: Caso 41 - Ta_010_20_05



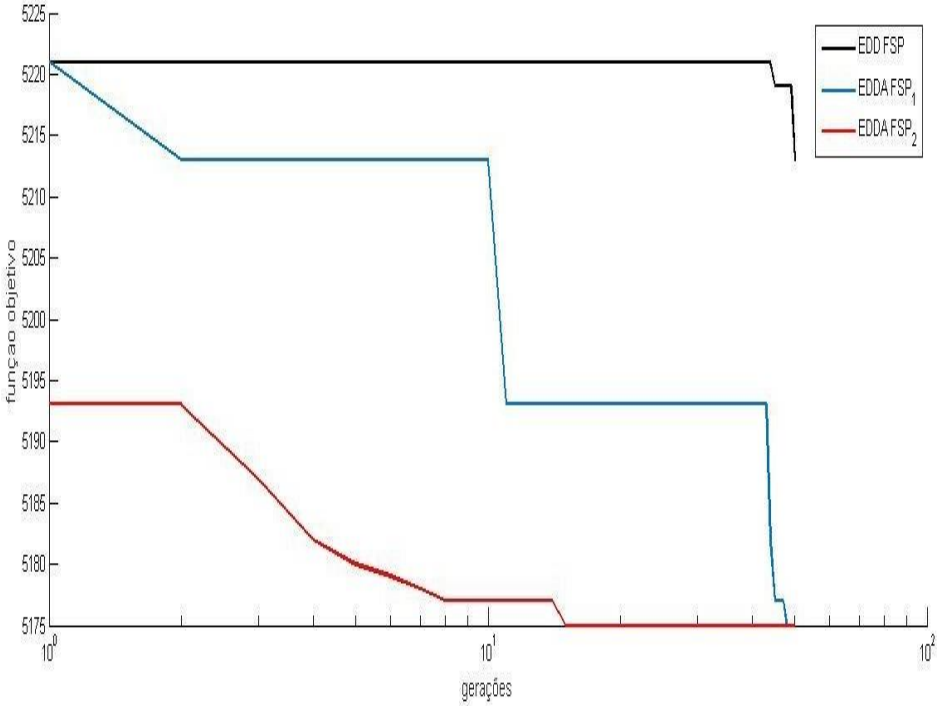
Fonte: o autor, 2017.

Figura 7.8 – Convergência: Caso 55 - Ta_024_20_20



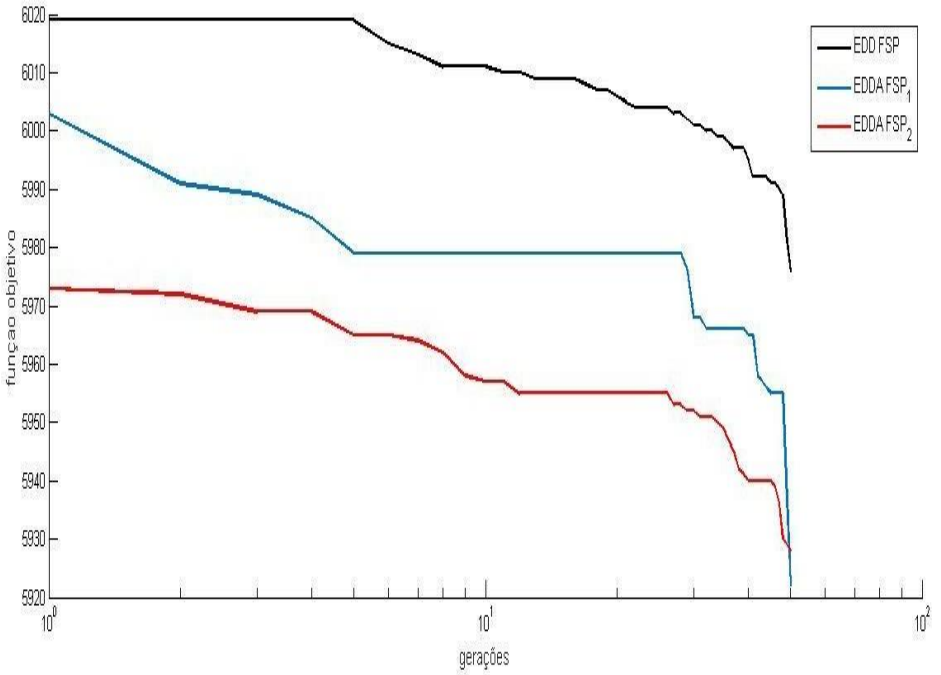
Fonte: o autor, 2017.

Figura 7.9 – Convergência: Caso 94 - Ta_063_100_05



Fonte: o autor, 2017.

Figura 7.10 – Convergência: Caso 110 - Ta_079_100_10



Fonte: o autor, 2017.

Para o caso 5 (Figura 7.1), em termos de convergência, o algoritmo EDDA-FSP_1 obteve desempenho superior aos algoritmos EDD-FSP e EDDA-FSP_2. A convergência do algoritmo EDDA-FSP_1 ocorreu em torno da 100ª geração, enquanto que a convergência do algoritmo EDD-FSP ocorreu em torno da 1200ª geração e a convergência do algoritmo EDDA-FSP2 ocorreu em torno da 700ª geração. Ressalta-se que os três algoritmos foram capazes de alcançar a melhor solução conhecida na literatura para o caso em questão.

Para o caso 9 o algoritmo EDDA-FSP_1 também obteve melhor desempenho que os algoritmos EDD-FSP e EDDA-FSP_2 em termos de convergência. Conforme pode ser visualizado na Figura 7.2, a convergência do algoritmo EDDA-FSP_1 ocorreu em torno da 100ª geração, enquanto que os algoritmos EDD-FSP e EDDA-FSP_2 convergiram em torno da 1500ª geração e da 1300ª geração, respectivamente. No entanto, para este caso teste, destaca-se que os algoritmos EDDA-FSP_1 e EDDA-FSP_1 foram capazes de obter valores para melhores para função objetivo, isto é, valores menores para o *Makespan,* enquanto que o algoritmo EDD-FSP alcançou a melhor solução conhecida na literatura.

Para o caso 10, ilustrado na Figura 7.3, o algoritmo EDDA-FSP_2 obteve melhor desempenho em termos de convergência, sendo que esta ocorreu em torno da 700ª geração. Para este caso, a convergência dos algoritmos EDD-FSP e EDDA-FSP_2 ocorreu em torno da 1550ª geração e da 1500ª geração, respectivamente. Para este caso teste, destaca-se que os algoritmos EDDA-FSP_1 e EDDA-FSP_1 foram capazes de obter valores melhores para função objetivo (menores *Makespan*), enquanto que o algoritmo EDD-FSP produziu soluções inferiores (maiores *Makespan*) a melhor solução conhecida na literatura.

Os algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 para os casos 21 e 31 apresentaram desempenhos semelhantes, pois conforme pode ser visualizado nas Figuras 7.4 e 7.5, os três algoritmos convergiram em torno da 1500ª geração. Para os casos 21 e 31 nenhum dos três algoritmos foi capaz de alcançar a melhor solução conhecida na literatura.

Para o caso 36, ilustrado na Figura 7.6, o algoritmo EDDA-FSP_1 obteve melhor desempenho em termos de convergência, sendo que esta ocorreu em torno da 100ª geração. Para este caso, a convergência dos algoritmos EDD-FSP e EDDA-FSP_2 ocorreu em torno da 1500ª geração, respectivamente. Para este caso teste,

destaca-se que somente o algoritmo EDDA-FSP_1 foi capaz de alcançar a melhor solução conhecida na literatura.

Para o caso 41, em termos de convergência, o algoritmo EDDA-FSP_2 obteve desempenho superior aos algoritmos EDD-FSP e EDDA-FSP_1. A convergência do algoritmo EDDA-FSP2, conforme pode ser visto na Figura 7.7, ocorreu em torno da 100ª geração, enquanto que a convergência do algoritmo EDD-FSP ocorreu em torno da 1500ª geração e a convergência do algoritmo EDDA-FSP1 ocorreu em torno da 600ª geração. Ressalta-se que os três algoritmos foram capazes de alcançar a melhor solução conhecida na literatura para o caso em questão.

Os algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 para o caso 55 apresentaram desempenhos semelhantes, pois conforme mostrado na Figura 7.8, os três algoritmos convergiram em torno da 1500ª geração. Para este caso, o algoritmo EDD-FSP não foi capaz de alcançar a melhor solução conhecida na literatura.

Para o caso 94, em termos de convergência, o algoritmo EDDA-FSP_2 obteve desempenho superior aos algoritmos EDD-FSP e EDDA-FSP_1. Conforme ilustrado na Figura 7.9, a convergência do algoritmo EDDA-FSP2 ocorreu em torno da 1100ª geração, enquanto que a convergência do algoritmo EDD-FSP ocorreu em torno da 1550ª geração e a convergência do algoritmo EDDA-FSP_1 ocorreu em torno da 1500ª geração. Ressalta-se que o algoritmo EDD-FSP não foi capaz de alcançar a melhor solução conhecida na literatura para este caso.

Os algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 para o caso 110 apresentaram desempenhos semelhantes, pois conforme pode ser visto na Figura 7.10, os três algoritmos convergiram entre a 1500ª e 2000ª geração. Para este caso, o algoritmo EDD-FSP não foi capaz de alcançar a melhor solução conhecida na literatura.

Para os casos testes selecionados para a análise da convergência dos algoritmos, verifica-se que o algoritmo EDDA-FSP_1 apresenta melhor desempenho para 3 dos 10 casos testados (30% dos casos), o algoritmo EDDA-FSP_2 apresenta melhor desempenho para outros 3 dos 10 casos testados (30% dos casos), enquanto que os algoritmos EDDA-FSP_1 e EDDA-FSP_2 apresentam desempenho semelhante em 4 dos 10 casos (40% dos casos).

De modo geral, nota-se que para determinados casos o algoritmo EDDA-FSP_1 apresentou melhor desempenho e que para outros casos o algoritmo EDDA-FSP_2 apresentou melhor desempenho, mas, fazendo-se uma análise de maneira

global, pode-se dizer que o desempenho dos algoritmos EDDA-FSP_1 e EDDA-FSP_2 foram parecidos. Não se observou uma clara vantagem, no geral, em termos de desempenho quando se compara os resultados obtidos por ambos os algoritmos.

7.4 TEMPOS COMPUTACIONAIS

Os tempos médios, em segundos, do processamento computacional de uma simulação, dos algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 para os grupos de casos testes dos *Benchmarks* de Carlier, Heller, Reeves e Taillard são sumarizados nas Tabelas 7.10 a 7.13.

Tabela 7.10 - Tempos Computacionais: Casos testes – *Benchmark* Carlier

Caso	Descrição	EDD-FSP	EDDA-FSP_1	EDDA-FSP_2
		Tempo (seg)	Tempo (seg)	Tempo (seg)
1	Car_001_11_05	2,78	1,42	4,72
2	Car_002_13_04	3,09	2,18	0,20
3	Car_003_12_05	3,34	1,99	0,03
4	Car_004_14_04	3,54	2,31	0,35
5	Car_005_10_06	3,67	1,66	0,05
6	Car_006_08_09	3,72	1,34	0,45
7	Car_007_07_07	3,59	1,08	6,02
8	Car_008_08_08	3,72	1,30	0,38

Fonte: o autor, 2017.

Tabela 7.11 - Tempos Computacionais: Casos testes – *Benchmark* Heller

Caso	Descrição	EDD-FSP	EDDA-FSP_1	EDDA-FSP_2
		Tempo (seg)	Tempo (seg)	Tempo (seg)
9	Hel_001_20_10	4,73	3,66	0,14
10	Hel_002_100_10	41,07	66,57	0,82

Fonte: o autor, 2017.

Tabela 7.12 - Tempos Computacionais: Casos testes – *Benchmark* Reeves

Caso	Descrição	EDD-FSP	EDDA-FSP$_1$	EDDA-FSP$_2$
		Tempo (seg)	Tempo (seg)	Tempo (seg)
11	Rec_001_20_05	3,01	2,85	0,41
12	Rec_003_20_05	3,25	2,85	0,21
13	Rec_005_20_05	3,57	2,94	2,22
14	Rec_007_20_10	4,27	5,22	0,69
15	Rec_009_20_10	4,41	5,15	1,75
16	Rec_011_20_10	4,59	5,29	1,39
17	Rec_013_20_15	5,18	6,47	3,22
18	Rec_015_20_15	5,90	6,31	0,73
19	Rec_017_20_15	5,25	6,28	0,47
20	Rec_019_30_10	5,70	7,19	0,98
21	Rec_021_30_10	5,63	9,68	0,50
22	Rec_023_30_10	5,76	9,69	0,04
23	Rec_025_30_15	6,64	9,10	1,35
24	Rec_027_30_15	5,15	12,26	0,39
25	Rec_029_30_15	5,07	12,19	0,23
26	Rec_031_50_10	5,33	17,04	0,21
27	Rec_033_50_10	5,78	16,88	1,58
28	Rec_035_50_10	6,35	22,89	1,28
29	Rec_037_75_20	12,90	68,62	1,36
30	Rec_039_75_20	13,48	83,32	10,37
31	Rec_041_75_20	13,21	73,62	0,72

Fonte: o autor, 2017.

Tabela 7.13 - Tempos Computacionais: Casos testes – *Benchmark* Taillard (continua...)

Caso	Descrição	EDD-FSP	EDDA-FSP$_1$	EDDA-FSP$_2$
		Tempo (seg)	Tempo (seg)	Tempo (seg)
32	Ta_001_20_05	4,72	0,61	1,07
33	Ta_002_20_05	4,25	0,71	0,13
34	Ta_003_20_05	4,22	0,75	0,40
35	Ta_004_20_05	4,32	0,80	0,07
36	Ta_005_20_05	4,24	0,88	0,63
37	Ta_006_20_05	4,16	0,54	0,13
38	Ta_007_20_05	4,21	0,65	1,18
39	Ta_008_20_05	4,21	0,60	0,21
40	Ta_009_20_05	4,31	0,85	0,13
41	Ta_010_20_05	4,30	0,65	1,46
42	Ta_011_20_10	5,16	0,64	0,68
43	Ta_012_20_10	4,81	0,64	1,52
44	Ta_013_20_10	4,83	0,83	2,11
45	Ta_014_20_10	4,90	0,65	0,56
46	Ta_015_20_10	4,81	0,66	2,84
47	Ta_016_20_10	4,76	0,90	0,17
48	Ta_017_20_10	4,81	0,55	4,67
49	Ta_018_20_10	5,27	0,66	3,41
50	Ta_019_20_10	4,98	0,64	5,28
51	Ta_020_20_10	4,98	0,84	0,36
52	Ta_021_20_20	6,14	0,59	0,10
53	Ta_022_20_20	6,07	0,73	0,07
54	Ta_023_20_20	4,42	0,70	0,07
55	Ta_024_20_20	4,37	0,73	0,27

Fonte: o autor, 2017.

Tabela 7.13 - Tempos Computacionais: Casos testes – *Benchmark* Taillard (...fim)

56	Ta_025_20_20	5,05	0,80	0,01
57	Ta_026_20_20	5,55	0,56	1,45
58	Ta_027_20_20	5,63	0,57	2,93
59	Ta_028_20_20	5,79	0,77	1,62
60	Ta_029_20_20	7,47	0,71	0,96
61	Ta_030_20_20	9,06	0,74	0,16
62	Ta_031_50_05	9,64	0,69	5,00
63	Ta_032_50_05	9,74	0,96	2,02
64	Ta_033_50_05	10,32	0,72	0,60
65	Ta_034_50_05	10,39	0,35	3,49
66	Ta_035_50_05	10,36	0,78	4,20
67	Ta_036_50_05	10,50	0,91	16,38
68	Ta_037_50_05	10,34	0,68	7,66
69	Ta_038_50_05	10,05	0,92	8,55
70	Ta_039_50_05	10,48	0,47	5,51
71	Ta_040_50_05	6,98	0,44	0,04
72	Ta_041_50_10	8,12	0,62	2,34
73	Ta_042_50_10	8,07	0,59	11,49
74	Ta_043_50_10	8,01	0,50	17,04
75	Ta_044_50_10	7,98	0,76	1,14
76	Ta_045_50_10	5,85	0,58	2,71
77	Ta_046_50_10	5,69	0,42	0,52
78	Ta_047_50_10	6,64	0,69	0,85
79	Ta_048_50_10	7,25	0,78	4,71
80	Ta_049_50_10	7,62	0,49	12,08
81	Ta_050_50_10	7,95	0,57	5,87
82	Ta_051_50_20	9,92	0,57	0,04
83	Ta_052_50_20	8,84	0,42	3,38
84	Ta_053_50_20	10,37	0,48	8,26
85	Ta_054_50_20	10,47	0,61	12,87
86	Ta_055_50_20	7,95	0,58	4,42
87	Ta_056_50_20	9,97	0,34	17,86
88	Ta_057_50_20	10,66	0,57	8,81
89	Ta_058_50_20	10,79	0,44	6,46
90	Ta_059_50_20	11,01	0,38	8,48
91	Ta_060_50_20	11,05	0,49	0,07
92	Ta_061_100_05	9,25	0,84	6,45
93	Ta_062_100_05	7,94	0,58	2,01
94	Ta_063_100_05	6,92	0,65	6,26
95	Ta_064_100_05	8,47	0,56	28,08
96	Ta_065_100_05	8,96	0,52	36,12
97	Ta_066_100_05	9,13	0,72	13,55
98	Ta_067_100_05	9,34	0,65	14,14
99	Ta_068_100_05	8,83	0,87	3,16
100	Ta_069_100_05	6,40	0,64	25,48
101	Ta_070_100_05	7,34	0,51	3,90
102	Ta_071_100_10	11,04	0,84	13,15
103	Ta_072_100_10	10,54	0,53	23,42
104	Ta_073_100_10	10,23	0,53	16,86
105	Ta_074_100_10	11,41	0,63	4,28
106	Ta_075_100_10	12,02	0,61	19,98
107	Ta_076_100_10	11,69	0,70	32,16
108	Ta_077_100_10	8,59	0,69	16,92
109	Ta_078_100_10	9,09	0,47	35,92
110	Ta_079_100_10	12,03	0,51	24,36

Fonte: o autor, 2017.

Em relação aos tempos médios computacionais, para os casos testes dos *Benchmarks* de Carlier, o algoritmo EDDA-FSP_2 apresentou os menores tempos computacionais para 6 dos 8 casos testados, enquanto que o algoritmo EDDA-FSP_1 apresentou os menores tempos computacionais para 2 dos 8 casos testados.

Para todos os casos testes dos *Benchmarks* de Heller e Reeves, o algoritmo EDDA-FSP_2 apresentou os menores tempos computacionais.

Para os casos testes do *Benchmark* de Taillard, o algoritmo EDDA-FSP_1 apresentou os menores tempos computacionais para 59 dos 79 casos testados, enquanto que o algoritmo EDDA-FSP_2 apresentou os menores tempos computacionais para apenas 20 dos 79 casos testados.

CONSIDERAÇÕES FINAIS

A presente tese tratou da aplicação de um paradigma da Computação Evolutiva (CE), que associa os princípios da teoria da evolução de *Darwin* e da seleção natural para a resolução de problemas de otimização complexos e engloba algoritmos de otimização estocástica, denominados Algoritmos Evolutivos (AEs), inspirados em fenômenos de hereditariedade, variação e seleção em um nível abstrato.

Devido a sua facilidade de implementação e eficiência na busca de soluções adequadas para problemas complexos de otimização, o algoritmo de Evolução Diferencial (ED), categorizado como AE, foi considerado neste estudo, cujo propósito geral foi o desenvolvimento de novos métodos de solução para Problemas de Programação da Produção (PPPs) baseados em abordagens de Evolução Diferencial.

Nesta tese, foram propostas para a minimização do *Makespan* em problemas de programação da produção em ambientes *Flow Shop* Permutacional, uma variante do algoritmo de Evolução Diferencial Discreta (EDD) e duas variantes do algoritmo de Evolução Diferencial Discreta Autoadaptativa (EDDA).

Dado que o algoritmo de ED foi originalmente projetado para trabalhar com variáveis contínuas, a forma de lidar com a conversão de variáveis contínuas em variáveis discretas, e vice-versa é uma importante questão tratada nas aplicações de algoritmos de ED em PPPs. Assim, o desenvolvimento de uma variante discreta do algoritmo de ED constitui uma das contribuições desta tese, pois possibilitou avaliar o comportamento do algoritmo de ED em relação à solução de PPPs em ambientes *Flow Shop* Permutacional, um problema de significância prática e acadêmica amplamente reconhecida.

Uma importante característica do algoritmo de ED é a pequena quantidade de parâmetros utilizados, sendo eles a ponderação da diferença empregada (F), a probabilidade de ocorrência de cruzamento (CR), a quantidade de indivíduos/vetores mantidos na população (Np) e o número de gerações realizadas durante o processo. No entanto, como em outros AEs, o desempenho da ED é bastante dependente da configuração dos parâmetros de controle, algumas versões do algoritmo de ED com parâmetros de controle adaptativos ou autoadaptativos têm sido propostas na literatura especializada.

Em uma revisão de literatura, que cobriu 40 trabalhos orientados ao desenvolvimento de métodos de solução baseados em abordagens de ED para os PPPs em ambientes *Flow Shop* Permutacional, foi possível identificar que poucos estudos adotam abordagens de ED autoadaptativas para a solução do problema em questão. Portanto, outra contribuição desta tese foi cobrir algumas lacunas da literatura, através do desenvolvimento de novos métodos baseados em ED, com ênfase nas abordagens autoadaptativas.

Nesta tese, os pressupostos que foram considerados para o desenvolvimento dos novos métodos de ED incluíram: Mecanismos autoadaptativos para a seleção da estratégia de ED e para a configuração dos parâmetros *F* e *CR*; e Regra LOV (*Largest-Order-Value*) para converter os vetores no domínio contínuo em vetores permutações de tarefas no domínio discreto.

Os algoritmos de Evolução Diferencial propostos nesta tese foram denominados Evolução Diferencial Discreta para *Flow Shop* Permutacional (EDD-FSP) e Evolução Diferencial Discreta Autoadaptativa para *Flow Shop* Permutacional (EDDA-FSP_1 e EDDA-FSP_2).

O algoritmo EDD-FSP, foi desenvolvido tendo como base a versão clássica do algoritmo de ED e a regra LOV (*Largest-Order-Value*) para conversão dos domínios contínuos em domínios discretos. O algoritmo EDDA-FSP_1 foi desenvolvido tendo como base o algoritmo JADE (*J Adaptive Differential Evolution*), em que uma nova estratégia de mutação *ED/current-to-p-best* é implementada e os parâmetros *F* e *CR* são ajustados autoadaptativamente durante o processo evolutivo. E o algoritmo EDDA-FSP_2, foi desenvolvido tendo como base o algoritmo SaDE (*Self-adaptive Differential Evolution*), em que a escolha da estratégia de ED e dois dos parâmetros de controle *F* e *CR* são ajustados autoadaptativamente durante o processo evolutivo.

Os algoritmos EDD-FSP, EDDA-FSP_1 e EDDA-FSP_2 foram executados em 110 casos testes, disponíveis nos *Benchmarks* de Carlier, Heller, Reeves e Taillard. O desempenho dos algoritmos foi avaliado em termos da qualidade das soluções fornecidas, por meio dos erros relativos percentuais, e em termos da capacidade de exploração do espaço de busca por meio do desvio padrão e da convergência.

No que diz respeito ao desempenho dos algoritmos em termos de erros relativos percentuais, o algoritmo EDDA-FSP_1 apresentou melhor desempenho para os casos testes do *Benchmark* de Taillard, enquanto que o algoritmo EDDA-FSP_2

apresentou melhor desempenho para os casos testes do *Benchmark* de Reeves. Para os casos testes dos *Benchmarks* de Carlier e de Heller os algoritmos EDDA-FSP_1 e EDDA-FSP_2 apresentaram o mesmo desempenho.

No que diz respeito ao desempenho dos algoritmos em termos de desvio padrão, o algoritmo EDDA-FSP_1 apresentou melhor desempenho para os casos testes dos *Benchmarks* de Reeves e de Taillard, enquanto que o algoritmo EDDA-FSP_2 apresentou melhor desempenho para os casos testes dos *Benchmarks* de Carlier e Heller.

Para os casos testes selecionados para a análise da convergência dos algoritmos, verifica-se que o algoritmo EDDA-FSP_1 apresenta melhor desempenho para 3 dos 10 casos testados (30% dos casos), o algoritmo EDDA-FSP_2 apresenta desempenho superior para outros 3 dos 10 casos testados (30% dos casos), enquanto que os algoritmos EDDA-FSP_1 e EDDA-FSP_1 apresentam desempenho semelhante em 4 dos 10 casos (40% dos casos).

Em relação aos tempos médios computacionais, o algoritmo EDDA-FSP2 apresentou os menores tempos computacionais para a maioria dos casos dos *benchmarks* de Carlier, Heller e Reeves, enquanto que o algoritmo EDDA-FSP1 apresentou os menores tempos computacionais para a maioria dos casos do *benchmark* de Taillard.

De forma geral os resultados mostram que as abordagens propostas nesta tese são promissoras para a solução de PPPs em ambientes FSP.

8.1 TRABALHOS FUTUROS

Em relação aos algoritmos propostos nesta tese, entre os possíveis trabalhos futuros destacam-se:

- Comparar o desempenho dos algoritmos propostos com outros Algoritmos Evolutivos e de Inteligência de Enxames, como por exemplo, o Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*);
- Incorporar, nos algoritmos propostos, mecanismos de busca local visando evitar estagnações e/ou melhorar a capacidade de exploração dos algoritmos; e

- Incorporar, nos algoritmos propostos, heurísticas para geração de uma parcela da população inicial com vistas a gerar diversidade e melhorar a qualidade das soluções iniciais.

Em relação ao desenvolvimento de métodos de solução para PPPs baseadas em abordagens de ED, entre os possíveis trabalhos futuros destacam-se:

- Desenvolver novo(s) algoritmo(s) baseado(s) em ED para solução de PPP em ambientes *Flow Shop* Permutacional problemas bicritério e multicritério;
- Verificar a aplicabilidade de algoritmo(s) baseado(s) em ED para solução de PPP em ambientes *Flow Shop* Permutacional, em problemas práticos e reais; e
- Desenvolver novo(s) algoritmo(s) baseado(s) em ED para solução de PPP em outros ambientes, como por exemplo, *Flow Shop* Híbrido.

8.2 PUBLICAÇÕES RELACIONADAS À TESE

As publicações relacionadas à tese foram as seguintes:

1) COELHO, L. S.; MARIANI, V. C.; VASCONCELOS SEGUNDO, E. H.; MORAIS, M. F.; FREIRE, R. Z. A zaslavskii firefly approach applied to loney´s solenoid *benchmark*. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), Vacation Road, San Diego, CA, USA: IEEE, 2014.

2) MORAIS, M. F.; BOIKO, T. J. P.; COELHO, L. S.; ROCHA, R. P.; PARAÍSO, P. R. Multicriteria hybrid flow shop scheduling problem: literature review, analysis and future research. **Independent Journal of Management & Production (IJM&P)**, v. 5, n.3, p. 1004-1031, 2014.

3) CARVALHO, L.; MORAIS, M. F.; COELHO, L. S.; ROCHA, R. P.; BELINE, E. L. Evolução diferencial: características dos métodos de solução para a programação da produção em ambientes flow shop permutacional. In: XXXVI

Encontro Nacional da Engenharia de Produção (ENEGEP), João Pessoa, Paraíba, PB, 2016.

4) HULTMANN AYALA, H. V.; KELLER, P.; MORAIS, M. F.; MARIANI, V. C.; COELHO, L. S.; RAO, R. V. Design of heat exchangers using a novel multiobjective free search differential evolution paradigm. **Applied Thermal Engineering**, v. 94, p. 170-177, 2016.

5) BOARETTO, M. A.R.; MORAIS, M. F.; COELHO, L. S. Evolução diferencial aplicada ao problema de programação da produção em sistemas flow shop permutacional. In: XLIX Simpósio Brasileiro de Pesquisa Operacional (SBPO), Blumenau, SC, 2017.

6) MORAIS, M. F.; BOARETTO, M. A.R.; COELHO, L. S.; ROCHA, R. P.; MOTTA, J. F. Evolução diferencial autoadaptativa para solução de problemas de programação da produção em flow shop permutacional. In: XXXVII Encontro Nacional da Engenharia de Produção (ENEGEP), Joinville, SC, 2017.

REFERÊNCIAS

ABBASS, H. A. The self-adaptive Pareto differential evolution algorithm. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2002), Honolulu, Hawai, Estados Unidos da América. **Anais…** Honolulu, Hawai, Estados Unidos da América: IEEE, 2002.

AHMADIZAR, F.; BARZINPOUR, F.; ARKAT, J. Solving permutation flow shop sequencing using ant colony optimization. In: IEEE INTERNATIONAL CONFERENCE ON. INDUSTRIAL ENGINEERING AND ENGINEERING MANAGEMENT, 2007. **Anais**... Singapore: IEEE, 2007.

AKROUT, H.; JARBOUI, B.; REBAI, A.; SIARRY, P. New greedy randomized adaptive search procedure based on differential evolution algorithm for solving no-wait flow shop scheduling problem. In: INTERNATIONAL CONFERENCE ON ADVANCED LOGISTICS AND TRANSPORT (ICALT 2013), 2013, Sousse, Tunisia. **Anais…** Sousse, Tunisia: IEEE, 2013a.

AKROUT, H.; JARBOUI, B.; REBAI, A.; SIARRY, P. A hybrid GRASP-Differential Evolution algorithm for solving flow shop scheduling problems with no-wait constraints. In: JABOUI, B.; SIARRY, P.; TEGHEM, J. **Metaheuristics for Production Scheduling.** Hoboken, USA: John Wiley & Sons, Inc., 2013b. Cap. 3, p. 45-68.

ALLAHVERDI, A. A survey of scheduling problems with no-wait in process. **European Journal of Operation Research,** v. 255, pp. 665-686, 2016.

ALLAHVERDI, A. The third comprehensive survey on scheduling problems with setup times/costs. **European Journal of Operation Research,** v. 246, pp. 345-378, 2015.

ALLAHVERDI, A.; CHENG, T. C. E.; KOVALYOV, M. Y. A survey of scheduling problems with *Setups* times or costs. **European Journal of Operational Research,** v. 187, n. 3, p. 985-1032, 2008.

AMIRIAN, H.; SAHRAEIAN, R. Multi-objective differential evolution for the flow shop scheduling problem with a modified learning effect. **International Journal of Engineering,** v. 27, n. 9, p. 1395-1404, 2014.

ANCĂU, M. On solving flowshop scheduling problems. **Proceedings of The Romanian Academy,** Series A, v. 13, n. 1, p. 71-79, 2012.

ANGELINE, P. J. Adaptive and self-adaptive evolutionary computations. In: PALANISWAMI, M.; ATTIKIOUZEL, Y.; II MARKS, R. J.; FOGEL, D. B.; FUKUDA, T. **Computational Intelligence: A Dynamic Systems Perspective**. Estados Unidos da América: IEEE Press, 1995.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z.. **Evolutionary Computation 2: Advanced Algorithms and Operators.** Oxford, Bristol, Reino Unido: Institute of Physics Publishing and Oxford University Press, 2000.

BAKER, K.R. **Introduction to sequencing and scheduling.** New York, Estados Unidos da América: John Wiley & Sons, Inc., 1974.

BANITALEBI, A.; AZIS, M. I. A.; AZIZ, Z. A. A self-adaptive binary differential evolution algorithm for large scale binary optimization problems**. Information Sciences**, v. 367-368, p. 487-511, 2016.

BEDWORTH, D. D.; BAILEY, J. E. **Integrated Production Control Systems: management, analysis, design.** 2ª ed. New York, Estados Unidos da América: John Wiley & Sons, Inc., 1987.

BINITHA, S.; SATHYA, S. S. A survey of bio inspired optimization algorithms. **International Journal of Soft Computing and Engineering**, v. 2, n. 2, p. 137-151, 2012.

BOIKO, T. J. P. **Métodos heurísticos para a programação em flow shop permutacional com tempos de setup separados dos tempos de processamento e independentes da sequencia de tarefas.** 2008. 207 f. Dissertação (mestrado). Engenharia de Produção. Universidade de São Paulo, Escola de Engenharia de São Carlos. São Carlos, SP, 2011.

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. **Information Sciences**, v. 237, p. 82–117, 2013.

BRANKE, J. **Evolutionary Optimization in Dynamic Environments.** New York, Estados Unidos da América: Springer Science+Business Media, 2002.

BREST, J.; BOSKOVIC, B.; GREINER, S.; ZUMER, V.; MAUCEC, M.S. Performance comparison of self-adaptive and adaptive differential evolution algorithms. **Soft Computation,** v. 11, n. 7, p. 617-629, 2007.

BREST, J.; GREINER, S.; BOSKOVIC, B.; MERNIK, M.; ZUMER, V. Self-adapting control parameters in differential evolution: a comparative study on numerical *Benchmark* problems. **IEEE Transactions on Evolutionary Computation,** v. 10, n. 6, p. 646– 657, 2006.

BROWNLEE, J. **Clever Algorithms: Nature-Inspired Programming Recipes.** 1 ed. Austrália: Lulu, 2011.

CARLIER, J. Ordonnancements a contraintes disjonctives. **Operations Research,** v. 12, p. 333-351, 1978.

CARVALHO, L.; MORAIS, M. F.; COELHO, L. S.; ROCHA, R. P.; BELINE, E. L. Evolução Diferencial: Características dos Métodos de Solução para a Programação da Produção em Ambientes Flow Shop Permutacional. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO (ENEGEP), 36. **Anais…** João Pessoa,PB: ABEPRO, 2016.

CREPINSEK, M.; LUI; S.; MERNIK, M. Exploration and exploitation in evolutionary algorithms: A survey. **ACM Computing Surveys**, v. 45, n. 35, 2013.

CZAIKOSKI, G. H.; URIO, P. R.; GONÇALVES, R. A.; ALMEIDA, C. P.; KUK, J. N.; VENSKE, S. M. Evolução Diferencial com ensemble de Operadores de Mutação em GPGPUs para o Despacho Econômico de Energia Elétrica. **RITA – Revista de Informática Teórica e Aplicada**, v. 23, n. 2, p. 10-32, 2016.

DAHAL, K. P.; TAN, K. C.; COWLING, P. I. **Evolutionary Scheduling.** Studies in Computational Intelligence, v. 49. Berlin, Alemanha: Springer-Verlag Berlin Heidelberg, 2007.

DAS, S.; ABRAHAM, A.; CHAKRABORTY, U. K,; KONAR, A. Differential evolution using a neighborhood-based mutation operator. **IEEE Transactions on Evolutionary Computation**, v.13, n. 3, p. 526-553, 2009.

DAS, S.; MULLICK, S. S.; SUGANTHAN, P. N. Recents advances in differential evolution – an updated survey. **Swarm and Evolutionary Computation,** v. 27, p. 01-30, 2016.

DAS, S.; SUGANTHAN, P. N. Differential evolution: A survey of the state-of-the-art. **IEEE Transactions on Evolutionary Computation**, v. 15, n. 1, p. 4-31, 2011.

DAVENDRA, D.; ZELINKA, I.; SENKERIK, R.; PLUHACEK, M. Complex network analysis of differential evolution algorithm applied to flowshop with no-wait problem. In: SYMPOSIUM ON DIFFERENTIAL EVOLUTION (SDE 2014). **Anais…** Orlando, FL, USA: IEEE, 2014.

DAVENDRA, R.; ONWUBOLU, G. Flow shop scheduling using enhanced differential evolution algorithm. In: EUROPEAN CONFERENCE ON MODELLING AND SIMULATION (ECMS), 21. **Anais...** Praga, República Checa: IEEE, 2007.

DENG, G.; GU, X. A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with *Makespan* criterion. **Computers & Operations Research,** v. 39, p. 2152-2160, 2012.

DONG, M. A discrete differential evolution with tabu list approach to no-wait flow-shop scheduling. Proceedings of the 2015 In: INTERNATIONAL CONFERENCE ON INFORMATION AND AUTOMATION (ICIA 2015). **Anais…** Lijang, China: IEEE, 2015.

EIBEN, A. E.; HINTERDING, R.; MICHALEWICZ, Z. Parameter control in evolutionary algorithms. **IEEE Transactions on Evolutionary Computation,** v. 3, n. 2, p. 124-141, 1999.

EIBEN; A. E.; SMITH, J. E. **Introduction to Evolutionary Computing.** 2ª. ed. Berlin, Alemanha: Springer-Verlag Berlin, Alemanha, 2015.

EMMONS, H.; VAIRAKTARAKIS, G. **Flow shop scheduling: theoretical results, algorithms, and applications.** New York: Springer Science + Business Media, 2013.

FAN, Q.; YAN, X. Self-adaptive differential evolution algorithm with discrete mutation control parameters, **Expert Systems with Applications**, v. 42, n. 3, p. 161-176, 2015.

FAN, Q.; YAN, X. Self-Adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. **IEEE Transactions on Cybernetics**, v. 46, n. 1, p. 219-232, 2016.

FAN, Q.; ZHANG, Y. Self-adaptive differential evolution algorithm with crossover strategies adaptation and its application in parameter estimation. **Chemometrics and Intelligent Laboratory Systems**, v. 151, p.164-171, 2016.

FEOKTISTOV**, V. Differential Evolution: Search of Solutions.** Serie Springer Optimization and Its Applications, v. 5. USA: Springer-Verlag, 2006.

FOGEL, L. J. Autonomous automata, **Industrial Research,** v. 4, p. 14-19, 1962.

FOGEL, L. J. **On the organization of intellect.** Dissertation (PhD) Universidade da Califórnia, Los Angeles, CA, USA, 1964.

FRENCH, S. **Sequencing and scheduling: an introduction to the mathematics of the job shop.** New York, NY: John Wiley & Sons, Inc., 1982.

FU, C. M.; JIANG, C.; CHEN, G. S.; LIU, Q. M. An adaptive differential evolution algorithm with an aging leader and challengers mechanism. **Applied Soft Computing**, v. 57, p. 60-73, 2017.

FUCHIGAMI, H. Y.; RANGEL, S. Métodos heurísticos para maximização do número de tarefas justin-time em flow shop permutacional. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 47. **Anais...** Porto de Galinhas,PE, 2015.

GÄMPERLE, R.; MULLER, S. D.; KOUMOUTSAKOS, P. A parameter study for differential evolution, In: GRMELA, A.; MASTORAKIS, N. E. **Advances in intelligent systems, fuzzy systems, evolutionary computation**. Cambridge, Reino Unido: WSEAS Press, 2002.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A guide to theory of NP-Completeness.** San Francisco, CA, USA: Freeman, 1979.

GASPAR-CUNHA, A.; ANTUNES, R.; TAKAHASHI, C. H. **Manual de computação evolutiva e metaheurística.** Belo Horizonte, MG, Brasil: Editora da UFMG (Universidade Federal de Minas Gerais) e Imprensa da Universidade de Coimbra, 2012.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning.** Nova York, Estados Unidos da América: Addison-Wesley, 1989.

GOYAL, S. K.; SRISKANDARAJAH, C. No-Wait Shop Scheduling: Computational Complexity and Approximate Algorithms, **Operations Research,** v. 25, n. 4, p. 220-244, 1988.

GRAHAM**,** R. L.**;** LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G. Optimization and Approximation in Deterministic Sequencing and Scheduling: A survey. **Annals of Discrete Mathematics,** v. 5, p. 287-326, 1979.

GUIMARÃES, F. G.; SILVA, R. C. P.; PRADO, R. S.; MAGELA NETO, O.; DAVENDRA, D. D. Flow shop scheduling using a general approach for differential evolution. In: ZELINKA, I.; SNASEL, I.; ABRAHAM, V. **Handbook of Optimization: From Classical to Modern Approach.** Berlin, Alemanha: Springer Berlin Publisher, 2013, series Intelligent Systems Reference Library, v.3.p. 597-614.

GUO, S-M.; YANG, C-C.; TSAI, J, S-H.; HSU, P-H. A self-optimization approach for l-shade incorporated with eigenvector-based crossover and successful-parent selecting framework on CED 2015 *Benchmark* set. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2015). **Anais…** Sendai, Japan: IEEE, 2015.

GUPTA, J. N.D.; STAFFORD JR., E. F. Flowshop scheduling research after five decades. **European Journal of Operational Research,** v. 169, p. 699-711, 2006.

HELLER, J. Some numerical experiments for an mj flow shop and its decision-theoretical aspects. **Operations Research**, v. 8, p. 178–184,1960.

HOLLAND, J. H. Outline for a logical theory of adaptive systems. **Journal of the Association for Computing Machinery,** v. 3, p. 297-314, 1962.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems.** Massachusetts, Michigan, Estados Unidos da América: The MIT Press, 1975.

HU, R.; MENG, X.; QIAN, B.; LI, K. A differential evolution approach for NTJ-NFSSP with SDSTs. In: HUANG, D-S.; MA, J.; JO, K-H.; GROMILA, M. M. **Intelligent Computing Theories and Applications.** Lecture Notes in Artificial Intelligence, v. 7390, Berlin, Alemanha: Springer-Verlag, 2012, p. 288-299.

ICSI – International Computer Science Institute (2017). **Differential Evolution (DE) for Continuous Function Optimization (an algorithm by Kenneth Price and Rainer Storn).** Web page. http://www1.icsi.berkeley.edu/~storn/code.html. Acessado em 2017-02-02.

JARBOUI, B.; SIARRY, P.; TEGHEM, J. **Metaheuristics for Production Scheduling.** Londres, Reino Unido: ISTE Ltda., 2013.

JOHNSON S. M.; MONTGOMERY D. C. **Operations research in production, planning, scheduling and inventory control.** New York, Estados Unidos da América: John Wiley & Sons, Inc., 1974.

JOHNSON, S. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quaterly**, v.1, n.1, p. 61-68, 1954.

KARIMI, N.; DAVOUDPOUR, H. A high performing metaheuristic for multi-objective flowshop scheduling problem. **Computers & Operations Research**, v. 52, p. 149-156, 2014.

KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection.** Massachusetts, Michigan, Estados Unidos da América: The MIT Press, 1992.

KOZA, J. R. Hierarchical genetic algorithms operating on populations of computer programs. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 11. **Anais…** San Mateo, CA, Estados Unidos da América: Morgan Kaufmann, 1989.

LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G.; SHMOYS, D. B. **Sequencing and Scheduling: Algorithms and Complexity.** Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, Holanda, 1989.

LI, X.; YIN, M. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. **Advances in Engineering Software**, v. 55, p. 10-31, 2013.

LICHTBLAU, D. Discrete optimization using Mathematica. In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS (SCI 2002), 6. **Anais…** Orlando, Florida, Estados Unidos da América: International Institute of Informatics and Systemics, 2002.

LICHTBLAU, D. Relative position indexing approach. IN: ONWUBOLU ,G. C.; DAVENDRA, D. **Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization**, of Studies in Computational Intelligence, v. 175, Berlin, Alemanha: Springer-Verlag, Berlin, 2009.

LIFL - Laboratoire d'Informatique Fondamentale de Lille. **Benchamarks for flowshop**. Disponível em http://www.lifl.fr/~liefooga/*Benchmark*s/*Benchmark*s/. Consultado em Fevereiro de 2016.

LIN, Q.; GAO, L.; LI, X.; ZHANG, C. A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. **Computers & Industrial Engineering,** v. 85, p. 437-446, 2015.

LINDEN. R. **Algoritmos Genéticos.** 3ª. ed. Rio de Janeiro, RJ, Brasil: Editora Ciência Moderna Ltda., 2012.

LIU, J.; LAMPINEN, J. A fuzzy adaptive differential evolution algorithm**. Soft Computation,** v. 9, n. 6, p. 448–462, 2005.

LIU, J.; LAMPINEN, J. Adaptive parameter control of differential evolution. In: INTERNATIONAL MENDEL CONFERENCE ON SOFT COMPUTING (MENDEL 2002), 8. **Anais…** Brno, República Checa: 2002.

LIU, W. An improved differential evolution for permutation flowshop scheduling problem with total flowtime criterion. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCE, ENGINEERING DESIGN AND MANUFACTURING INFORMATIZATION (ICSEM), 3. **Anais…** Chengdu, China: IEEE, 2012.

LIU, Y.; YIN, M.; GU, W. An effective differential evolution algorithm for permutation flow shop scheduling problem. **Applied Mathematics and Computation,** v. 248, p. 143-159, 2014.

LIU, Y-F; LIU, S-Y. A hybrid discrete artificial bee colony algorithm for permutation flow-shop scheduling problem. **Applied Soft Computing,** v.13, n.3, p.1459-1463, 2013.

LOBATO, F. S.; GEDRAITE, R.; NEIRO, S. M. S. Solution of flow shop scheduling problems using the differential evolution algorithm. In: INTERNATIONAL CONFERENCE ON ENGINEERING OPTIMIZATION (EngOpt 2012), 3. **Anais…** Rio de Janeiro, RJ, Brasil: E-papers Serviços Editoriais Ltda., 2012.

LOPES, H. S.; TAKAHASHI, R. H. C. **Computação Evolucionária em Problemas de Engenharia.** Curitiba, PR, Brasil: Editora Omnipax Ltda, 2011.

MACCARTHY, B. L.; LIU, J. Y. Adressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research,** v. 31, n.1, p. 59-79, 1993.

MAHDAVI, S.; SHIRI, M. E.; RAHNAMAYAN, S. Metaheuristics in large-scale global continues optimization: A survey. Information Sciences, v. 295, p. 407-428, 2015.

MALLIPEDDI, R.; SUGANTHAN, P. N.; PAN, Q. K.; TASGETIREN, M. F. Differential evolution algorithm with ensemble of parameters and mutation strategies. **Applied Soft Computing**, v. 11, n. 2, p. 1679–1696, 2011.

MESQUITA, M.; COSTA, H. G.; LUSTOSA, L.; SILVA, A. S. M. **Programação detalhada da produção.** In: LUSTOSA, L.; MESQUITA, M. A.; QUELHAS, O.; OLIVEIRA, R. Planejamento e controle da produção. Rio de Janeiro, RJ: Elsevier, 2008.

MEZURA-MONTES, E.; VELÁZQUEZ-REYES , J.; COELLO, C.A.C. A comparative study of differential evolution variants for global optimization, in: **Proceedings** of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, 2006, pp. 4 85–4 92 .

MOKHTARI, H.; ABADI. I. N. K.; CHERAGHALIKHAMI. A. A multi-objective flow shop scheduling with resource-dependent processing times: trade-off between *Makespan* and cost of resources. **International Journal of Production Research,** v. 49, n. 19, p. 5851–5875, 2011.

MONTOYA, C.; MEJIA, G. Heuristic Algorithm for Workforce Scheduling Problems. **Brazilian Journal of Operations & Production Management**, v. 3, n. 2, p.35-48, 2006.

MORAIS, M. F.; MOCCELLIN, J. V. Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção *flow shop* híbridos com tempos de setup dependentes da sequência. **Gestão & Produção,** v. 17, n. 2, p. 367-375, 2010.

MORTON, T. E.; PENTICO, D. W. **Heuristic Scheduling Systems.** New York, NY, USA: John Wiley & Sons, Inc., 1993.

NAGANO, M. S.; BRANCO, F. J. C.; MOCCELLIN, J. V. Soluções de alto desempenho para a programação da produção flow shop. **Gestão da Produção, Operações e Sistemas – GEPROS**, v. 4, n. 2, p. 11-23, 2009.

NERI, F.; TIRRONEN, V. Recent advances in differential evolution: a survey and experimental analysis. **Artificial Intelligence Review**, v.33, p.61-106, 2010.

NORMAN, N.; IBA, H. Accelerating differential evolution using an adaptive local search. **IEEE Transactions on Evolutionary Computation**, v.12, n. 1, p. 107-125, 2008.

ONWUBOLU, G. Optimisation using differential evolution algorithm. **Technical Report**, TR- 2001-05. The University of the South Pacific. Institute of Applied Science (IAS), Suva, Fiji, 2001.

ONWUBOLU, G.; DAVENDRA, D. Scheduling flow shops using differential evolution algorithm. **European Journal of Operational Research**, v. 171, p. 647-692, 2006.

PAN, Q-K.; WANG, L.; QIAN, B. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. **Computers & Operations Research,** v. 36, p. 2498-2511, 2009,

PAN, Q-K.; TASGETIREN, M. F.; LIANG, Y-C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. **Computers & Industrial Engineering,** v. 55, p. 795-816, 2008.

PAN, Q-K.; WANG, L. A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems. **European Journal of Industrial Engineering,** v. 02, n. 03, p. 279-297, 2008,

PAN, Q-K.; WANG, L.; GAO, L.; LI, W. D. An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers**. Information Sciences,** v. 181, p. 668-685, 2011.

PAN, Q-K; TASGETIREN, M. F.; LIANG, Y-C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. In: CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION (GECCO'07). **Anais…** Londres, Inglaterra, Reino Unido: ACM, 2007.

PARK, J.; JEONG, Y.; LEE, W. An improved particle swarm optimization for economic dispatch problems with non-smooth cost functions. In: IEEE POWER ENGINEERING SOCIETY GENERAL MEETING. **Anais...** Montreal, Quebec, Canadá: IEEE, 2006.

PARKER, R. G. **Deterministic scheduling theory**. Londres, Reino Unido: Champman & Hall, 1995.

PEÑUÑURI, F.; CAB, C.; CARVENTE, O.; ZAMBRANO-ARJONA, M. A.; TAPIA, J. A. A study of the classical differential evolution control parameters. **Swarm and Evolutionary Computation**, v. 26, p. 86-96, 2016.

PIEREZAN, J.; FREIRE, R. Z.; WEIHMANN, L.; REYNOSO-MEZA, G.; COELHO, L. S. Static force capability optimization of humanoids robots based on modified self-adaptive differential evolution. **Computers & Operations Research**, v. 84, p. 205-215, 2017.

PINEDO, M. **Scheduling: theory, algorithms, and systems.** 3ª. ed. New Jersey, Estados Unidos da América: Prentice-Hall, 2008.

PRADO, R. S.; SILVA, R. C. P.; GUIMARÃES, F. G.; MAGELA NETO, O. (2010) Uma nova abordagem para a evolução diferencial em otimização discreta. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 18. **Anais...** Bonito, MS: CBA, 2010.

PRICE, K. V.; STORN, R. M.; LAMPINEN, J. A. **Differential evolution - A practical approach to global optimization.** Berlin, Alemanha: Springer-Verlag, 2005.

QIAN, B.; WANG, L.; HUANG, D-X.; WANG, X. Multi-objetive flow shop scheduling using differential evolution. In: HUANG, D-E.; IRWIN, G. W. **Intelligent Computing in Signal Processing and Pattern Recognition.** Lecture Notes in Control and Information Sciences, Alemanha: Springer-Verlag Berlin Heidelberg, 2006, p. 1125-1136.

QIAN, B.; WANG, L.; HU, R.; WANG, W-L.; HUANG, D-X.; WANG, X. A hybrid differential evolution method for permutation flow-shop scheduling. **International Journal Advanced Manufacturing Technology,** v. 38, p. 757-777, 2008.

QIAN, B.; WANG, L.; HU, R.; HUANG, D. X.; WANG, X. A DE-based approach to no-wait flow-shop scheduling. **Computers & Industrial Engineering,** v. 57, p. 787-805, 2009a.

QIAN, B.; WANG, L.; HUANG, D-X.; WANG, W-L.; WANG, X. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. **Computers & Operations Research,** v. 36, p. 209-233, 2009b.

QIAN, B.; WANG, L.; HUANG,D. X.; WANG, X. An effective hybrid DE-based algorithm for flow shop scheduling with limited buffers. **International Journal of Production Research,** v. 47, n. 1, p. 01-24, 2009c.

QIAN, B.; DU, P.; HU, R.; CHE, G. A differential evolution algorithm with two speed-up methods for NFSSP with SDSTs and Rds. In: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (WCICA), 10, Beijing, China. **Anais…** Beijing, China: IEEE, 2012a.

QIAN, B.; ZHOU, H-B.; HU, R.; FENG-HONG XIANG, F-H. Hybrid differential evolution optimization for no-wait flow-shop scheduling with sequence-dependent setup times and release dates. In: HUANG, D-S.; GAN, Y.; BEVILACQUA, V.; FIGUEROA, J. C. **Advanced Intelligent Computing**. 7th International Conference, ICIC 2011, Zhengzhou, Revised Selected Papers: 2012b.

QIAN, B.; WANG, J.; LIU, B.; HU, R.; CHE, G-L. DE-based algorithm for reentrant permutation flow-shop scheduling with different job reentrant times. In: SIMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN SCHEDULING (SCIS 2013), Singapura. **Anais…** Singapura: IEEE, 2013.

QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. **IEEE Transactions on Evolutionary Computation,** v. 13, n. 2, p. 398-417, 2009.

QIN, A. K.; SUGANTHAN, P. N. Self-adaptive differential evolution algorithm for numerical optimization. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2005), Edinburgh, Escócia, Reino Unido. **Anais…** Edinburgh, Escócia, Reino Unido: IEEE, 2005.

QING, A.; LEE, C. K. **Differential evolution in electromagnetics.** Berlin, Alemanha: Springer-Verlag, 2010.

RAMOS, A.; TUPIA, M. A systematic for solving flow shop scheduling problem using differential evolution algorithm. **International Journal of Applied Science and Technology,** v. 3, n. 7, p. 64-74, 2013.

RECHENBERG, I. Cybernetic solution path of an experimental problem. **Library Translation,** n. 1122, Royal Aircraft Establishment, Inglaterra, Reino Unido 1965.

REEVES, C. A genetic algorithm for flowshop sequencing. **Computers and Operations Research,** v. 22, p. 5–13, 1995.

ROGALSKY, T.; DERKSEN, R. W.; KOCABIYIK, S. Differential evolution in aerodynamic optimization. In: ANNUAL CONFERENCE OF CANADIAN AERONAUTICS AND SPACE INSTITUTE, 46. **Anais…** Montreal, Quebec, Canada: CASI, 1999.

RÖNKKÖNEN, J.; KUKKONEN, S.; PRICE, K. V. Real parameter optimization with differential evolution. In: IEEE Congress on Evolutionary Computation (CEC 2005). **Anais…** Edinburgh, Scotland: IEEE, 2005.

ROSÁRIO, R. R. L. **Algoritmos Evolutivos Adaptativos para Problemas de Programação de Pessoal.** 2011. 231f. Tese (Doutorado). Engenharia de Produção, Universidade Federal de Santa Catarina. Florianópolis, SC. 2011.

SALMAN, A. A.; AHMAD, I.; OMRAN, M. G. H; MOHAMMAD, M. G. Frequency assignment problem in satellite communications using differential evolution. **Computers & Operations Research,** v. 37, n. 12, p. 2152-2163, 2010.

SANTUCI, V.; BAIOLETTI, M.; MILANI, A. A Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem with Total Flow Time Criterion. In: ITALIAN WORKSHOP ON PLANNING AND SCHEDULING (IPS 2015), 6, Ferrara, Italy. **Anais…** Ferrara, Italy: CEUR-WS, 2015.

SANTUCI, V.; BAIOLETTI, M.; MILANI, A. Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. **IEEE Transactions on Evolutionary Computation**, v. 20, n. 5, 2016.

SARUHAN, H. Differential evolution and simulated annealing algorithms for mechanical systems design. **Engineering Science and Technology, an International Journal,** v. 17, n. 13, p. 131-136, 2014.

SAVSANI, P.; SAVSANI, V. Passing vehicle search (PVS): A novel metaheuristic algorithm. **Applied Mathematical Modelling**, v. 40, p. 3951-3978, 2016.

SAYADI, M. K.; RAMEZANIAN, R.; GHAFFARI-NASAD, N. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. **International Journal of Industrial Engineering Computations**, v. 1, p. 1-10, 2010.

SCHWEFEL, H. P. **Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik.** Diplomarbeit, Hermann Föttinger Institut für Strömungstechnik, Technische Universität, Berlin, Germany, 1965.

SHAO, W.; PI, D. A self-guided differential evolution with neighborhood search for permutation flow shop scheduling. **Expert Systems with Applications**, n. 51, p. 161-176, 2016.

SHEN, J-N.; WANG, L.; WANG, S-Y. A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. **Knowledge-Based Systems**, v. 74, p. 167-175, 2015.

SIMON, D. **Evolutionary optimization algorithms: biologically-inspired and population-based approaches to computer intelligence.** New Jersey, Estados Unidos da América: John Wiley & Sons, Inc., 2013.

SIPSER, M. **Introduction to the theory of computation**. Boston, Estados Unidos da América: Cengage Learning, 2012.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da produção.** 2 ed. São Paulo, Brasil: Atlas, 2002.

STORN, R. Differential evolution research - trends and open questions. In: CHAKRABORTY, U. K. **Advances in Differential Evolution.** Berlin, Alemanha: Springer-Verlag, 2008.

STORN, R.; PRICE, K. **Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces.** Technical Report TR-95-012, International Computer Science Institute. Berkeley, CA, Estados Unidos da América: 1995.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous space. **Journal of Global Optimization,** v. 11, p. 341-359, 1997.

SUGANTHAN, P. N. Differential evolution algorithm: Recent advances. In: DEDIU, A.-H.; MARTÍN-VIDE, C.; TRUTHE, B. **Theory and Practice of Natural**

Computing. Lecture Notes in Computer Science, v. 7505, p. 30-46. Berlin, Alemanha: Springer-Verlag, 2012.

SUN, Y.; ZHANG, C.; GAO, L.; WANG, X. Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. **The International Journal of Advanced Manufacturing,** v. 55, p. 723-739, 2011.

T´KINDT, V.. BILLAUT, J. C. **Multicriteria scheduling problems.** In: Ehrgott, M.; Gandibleux, X. Multiple criteria optimization: state of the art annotated bibliography surveys. Massachusetts,USA: Kluwer Academic Publishers, 2002.

TAILLARD, E. *Benchmark*s for basic scheduling problems. **European Journal of Operations Research,** n. 64, v. 2, p. 278-285, 1993.

TAILLARD, E. **Scheduling instances.** Disponível em http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html.

TANABE, R.; FUKUNAGA, A. Success-history based parameter adaptation for differential evolution. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2013). **Anais...** Cancún, México: IEEE, 2013.

TANABE, R.; FUKUNAGA, A. Improving the search performance of SHADE using linear population size reduction In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2014). **Anais...** Beijing, China: IEEE, 2014.

TASGETIREN, F.; CHEN,A.; GENCYILMAZ, G.; GATTOUFI, S. Smallest position value approach. IN: ONWUBOLU ,G. C.; DAVENDRA, D. **Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization**, of Studies in Computational Intelligence, v. 175, Berlin, Alemanha: Springer-Verlag, 2009.

TASGETIREN, M. F.; PAN, Q-K., SUGANTHAN, N.; CHEN, A. H-L. A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2010). **Anais…** Barcelona, Espanha: IEEE, 2010.

TASGETIREN, M. F.; PAN, Q-K.; KIZIALAY, D.; SUER, G. A populated local search with differential evolution for *Blocking* flowshop scheduling problem. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2015). **Anais…** Sendai, Japão: IEEE, 2015.

TASGETIREN, M. F.; PAN, Q-K.; SUGANTHAN, P. N.; BUYUKDAGLI, O. A variable iterated greedy algorithm with differential evolution for solving no-idle flowshops. In: RUTKOWSKI, L.; KORYTKOWSKI, M.; SCHERER, R.; TADEUSIEWICZ,

R.; ZADEH, L. A.; ZURADA, J. M. **Swarm and Evolution Computation.** Lecture Notes in Computer Science, v. 6838, Berlin, Alemanha: Springer-Verlag, 2012b, p. 128-135.

TASGETIREN, M. F.; PAN, Q-K.; SUGANTHAN, P. N.; BUYUKDAGLI, O. A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. **Computers & Operations Research,** v. 40, p. 1729-1743, 2013.

TASGETIREN, M. F.; PAN, Q-K.; SUGANTHAN, P. N.; CHUA, T. J. A differential evolution algorithm for no-idle flowshop scheduling problem with total tardiness criterion. **International Journal of Production Research,** v. 49, n. 16, p. 5033-5050, 2011a.

TASGETIREN, M. F.; PAN, Q-K.; SUGANTHAN, P. N.; CHEN, A. H-L. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. **Information Sciences**, v. 181, p. 3459–3475, 2011b.

TASGETIREN, M. F.; PAN, Q-K.; SUGANTHAN, P. N.; LIANG, Y-C. A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion. In: IEEE Symposium on Computational Intelligence in Scheduling (CISChed 2007). **Anais...** Honolulu, Hawai, Estados Unidos da América: IEEE, 2007.

TASGETIREN, M. F.; PAN, Q-K.; WANG, L.; CHEN, A. H-L.; A DE based variable iterated greedy algorithm for the no-idle permutation flowshop scheduling problem with total flowtime criterion. In: HUANG, D-S.; MA, J.; JO, K-H.; GROMILA, M. M. **Intelligent Computing Theories and Applications.** Lecture Notes in Artificial Intelligence, v. 6839, Berlin, Alemanha: Springer-Verlag, 2012a, p. 83-90.

TASGETIREN, M. F.; SEVIKLI, M.; LIANG, Y-C.; GENCYLMAZ, G. Differential evolution algorithm for permutation flowshop sequencing problem with *Makespan* criterion. Proceedings of the 4th In: INTERNATIONAL SYMPOSIUM ON INTELLIGENT MANUFACTURING SYSTEMS (IMS2004), 4. **Anais...** Sakarya, Turquia: IEEE, 2004.

TEO, J. Exploring dynamic self-adaptive populations in differential evolution. **Soft Computation,** v. 10, n. 8, p. 673–686, 2006.

TIEN, C-H.; HSU, C-Y.; CHEN, M-H.; CHANG, P-C. Differential evolutionary algorithms with novel mutation operator for solving the permutation flowshop

scheduling problem. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND ROBOTICS (ICCAR 2015). **Anais…** Singapura: IEEE, 2015.

TONGE, V. G.; KULKARNI, P. Solving permutation flowshop scheduling problem using improved differential evolutionary algorithm. **International Journal of Engineering Research & Technology,** v. 2, n. 10, p. 456-461, 2013.

TONGE, V. G.; KULKARNI, P. S. Permutation flowshop scheduling problem using DE: A survey. **International Journal of Societal Applications of Computer Science,** v. 01, n. 01, p. 39-41, 2012.

VÁZQUEZ-OJEDA, M; SEGOVIA-HERNÁNDEZ, J. G.; HERNÁNDEZ, S.; HERNÁNDEZ-AGUIRRE, A.; KISS, A. A. Optimization of an ethanol dehydration process using differential evolution algorithm. **Computer Aided Chemical Engineering,** v. 32, p. 217-222, 2013.

VIGNER, A.; BILLAUT, J-C.; PROUST, C. Solving k-stage hybrid flowshop scheduling problems. In: MULTICONFERENCE ON COMPUTATIONAL ENGINEERING SYSTEMS APPLICATIONS (CESA'99). **Anais…** New Jersey, USA, 1999.

WANG, H-Y.; LU, Y-B.; PENG, W-L. Permutation flow shop scheduling using a hybrid differential evolution algorithm. **International Journal of Computing Science and Mathematics,** v. 4, n. 3, p. 298-307, 2013.

WANG, L.; PAN, Q-K.; SUGANTHAN, P. N.; WANG, W-H.; WANG, Y-M. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. **Computers & Operations Research,** v. 37, p. 509-520, 2010.

WANG, Y.; CAI, Z.; ZHANG, Q. Differential evolution with composite trial vector generation strategies and control parameters. **IEEE Transactions on Evolutionary Computation**, v. 15, n. 1, p. 55-66, 2011.

WANG, Y.; CAI, Z.; ZHANG, Q. Enhancing the search ability of differential evolution through orthogonal crossover. **Information Sciences**, v. 185, n. 1, p. 153-177, 2012.

WANG, Y.; LI, H. X.; HUANG, T.; LI, L. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. **Applied Soft Computing**, v. 18, p. 232-247, 2014.

WARI, E.; ZHU, W. A survey on metaheuristics for optimization in food manufacturing industry. **Applied Soft Computing**, v. 46, p. 328-343, 2016.

WOLPERT, D.H.; MACREADY, W.G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67-82, 1997.

XU, X.; XIANG, Z.; WANG, W. A self-adaptive differential evolution for the permutation flow shop scheduling problem. In: Chinese Control and Decision Conference (CCDC, 2010). **Anais...** Xuzhou, China: IEEE, 2010.

YANG, W.; LIAO, C. Survey of scheduling research involving setup times. **International Journal of Systems Science,** v. 30, n. 2, p. 143-155, 1999.

YANG, X-S. **Engineering Optimization: An Introduction with Metaheuristic Applications.** New York, Estados Unidos da América: John Wiley & Sons, Inc., 2010.

YANG, Z.; TANG, K.; YAO, X. Self-adaptive differential evolution with neighborhood search. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2008). **Anais...** Hong Kong, China: IEEE, 2008.

YENISEY, M. M.; YAGMAHAN, B. (2014) Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. **Omega**, v. 45, p. 119-135, 2014.

ZAHARIE, D. Influence of crossover on the behavior of differential evolution algorithms. **Applied Soft Computing**, v. 9, p. 1126-1138, 2009.

ZAMUDA, A.; BREST, J. Self-adaptive control parameters' randomization frequency and propagations in differential evolution. **Swarm and Evolutionary Computation**, v. 25, p. 72-99, 2015.

ZHANG, J.; SANDERSON, A. C. **Adaptive differential evolution: a robust approach to multimodal problem optimization.** Berlin, Alemanha: Springer-Verlag, 2009a.

ZHANG, J.; SANDERSON, A. C. JADE: adaptive differential evolution with optional external archive. **IEEE Transactions on Evolutionary Computation,** v. 13, n. 5, p. 945-958, 2009b.

ZHANG, J.; SANDERSON, A. C. JADE: Self-adaptive differential evolution with fast reliable convergence performance. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2007). **Anais...** Singapura: IEEE, 2007.

ZHENG, T.; YAMASHIRO, M. Solving flow shop scheduling problems by quantum differential evolutionary algorithm. **International Journal Advanced Manufacturing Technology,** v. 49, p. 643-662, 2010.

ZHOU, A.; QY, B-Y.; LI, H.; ZHAO, S-Z.; SUGANTHAN, P. N.; ZHANG, Q. Multiobjective evolutionary algorithms: a survey of the state of the art. **Swarm and Evolutionary Computation,** v. 1, p. 32-49, 2011.

APÊNDICE A - Características dos trabalhos que apresentam abordagens de Evolução Diferencial aplicadas aos Problemas de Programação da Produção em *Flow Shop* Permutacional

Referência	Características do PPP	Características do(s) Algoritmo(s) de ED
Tasgetien et al. (2004)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 2*n*; - Taxa de Mutação (F): 0,4; - Taxa de Cruzamento (CR): 0,5; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número máximo de gerações: 50; - Número de Runs: 10; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: GA e PSO; - Outras Características: Regra SPV.
Qian et al. (2006)	- Função Objetivo: Bicritério; - Critério de Desempenho: *Makespan* e *Tardiness*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 2*n*; - Taxa de Mutação (F): {0,2, 0,7}; - Taxa de Cruzamento (CR): {0,1, 0,2, 0,4, 0,8, 1,0}; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número máximo de gerações: 300; - Número máximo de runs: 20; - Hibridizações: VNS aplicado a somente 1/5 da população em cada geração; - *Benchmark* Utilizado: Carlier e Reeves; - Métodos Utilizados para Comparação: RWGA; .- Outras Características: Regra LOV.

Pan, Tasgetiren e Liang (2007)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 20; - Taxa de Mutação (F): 0,2; - Taxa de Cruzamento (CR): 0,8; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo Máximo Computacional (T_{max} = *n(m/2)t*, com t = 30, 60 e 90 milissegundos); - Número Máximo de Gerações----- ; - Número Máximo de Avaliações da Função-Objetivo ----- ; - Número de Runs: 5; - Hibridizações: RLS e IG; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: IG_RSLS; - Outras Características: ----.
Tasgetien et al. (2007)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flow Time*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurísticas NN e NEH; - Tamanho da População: n; - Taxa de Mutação (F): 0,8; - Taxa de Cruzamento (CR): 0,8; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Numero Máximo de Gerações: 1000; - Número Máximo de Runs: 5; - Hibridizações: VND (Variable Neighborhood Descent); - *Benchmark* Utilizado: Taillard tratado como Flow Shop No-Wait; - Métodos Utilizados para Comparação: SA, TS (Chins), TS (Pilot) e DPSO; - Outras Características: Emprega o operador de cruzamento PTL de dois cortes proposto por Pan et al. (2005).

Pan e Wang (2008)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: 20; - Taxa de Mutação (F): 0,8 - Taxa de Cruzamento (CR): 0,8 - Procedimento de Seleção: Individuo com menor valor da função-objetivo; - Critério máximo de Parada: Tempo máximo computacional (T_{max} = 10*n*, 20*n*, ..., 150*n* ms, T_{max} = 10*n*/1000 se e T_{max} = 150*n*/1000 s); - Número Máximo de Gerações-----; - Número Máximo de execuções/Runs: 5; - Hibridizações: Busca local; - *Benchmark* Utilizado: Taillard tratado como Flow Shop No-Wait; - Métodos Utilizados para Comparação: IG, KK, NEH modificada para No-Wait, DEVNS, PSOVNS e DDELS. - Outras Características: Método de aceleração.
Pan, Tasgetiren e Liang (2008)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flow Time*; e *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: {10, 200, 2*n*}; - Taxa de Mutação (F): 0,3; - Taxa de Cruzamento (CR): 0,8; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações e Tempo Máximo Computacional (T_{max} = (n(m/2)t, com t = 60 ms) - Número Máximo de Gerações: 500; - Número de Runs-----; - Hibridizações: RLS e IGA; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: DDE, PSO, CPSO, GA_REEV, CPSO_PNEH, IG_{RIS}, DDE_{RLS}, PSO_{VNS}, H-CPSO, NEHT, GA_RMA, HGA_RMA, SA_OP, SPIRIT, GA_CHEN, GA_REEV, GA_MIT, ILS, GA_AA, M-MMAS, PACO, IG_RS, IG_RSLS , IG, IGRIS , DDE, DDERLS); - Outras Características-----;

Qian et al. (2008)	- Função Objetivo: Monocritério e Bicritério; - Critério de Desempenho: Monocritério – *Makespan*; Bicritério – *Makespan* e *Tardiness*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 2*n*; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): {0,1, 0,2, 0,4, 0,6, 0,8, 1,0}; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: 300 e 300*n*(*n*-1); - Número Máximo de Runs: 20; - Hibridizações: Busca local; - *Benchmark* Utilizado: Carlier e Reeves; - Métodos Utilizados para Comparação: 3 variantes do método proposto HDE (HDE_NOL, HDE_ML e HDE_BL), ODE, NEH, PGA, SA e HGA, para o caso de função-objetivo monocritério. Para o caso de função-objetivo multicritério, o método proposto foi comparado com o método IMMOGLS2; - Outras Características: Regra heurística LOV.
Pan, Wang e Qian (2009)	- Função Objetivo: Bicritério; - Critério de Desempenho: *Makespan* e *Tardiness*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística PWQ; - Tamanho da População: 5*n*; - Taxa de Mutação (F): 0,2; - Taxa de Cruzamento (CR): 0,2; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo de execução (T_{max} = 10*mn* ms); - Número Máximo de Gerações----- ; - Número Máximo de Runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Carlier, Heller e Reeves; - Métodos Utilizados para Comparação: HDE e IMMOGLS2; - Outras Características: Operadores de mutação e cruzamento são desenvolvidos.

Qian et al. (2009a)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/target-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): {0,1, 0,8}; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: 300; - Número Máximo de Runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Carlier, Reeves e Heller; - Métodos Utilizados para Comparação: HDE_NOL, HDE_nospeed-up, HDE_NFNE, RAJ, HDE com esquemas de busca local, HDE_KS5, HDE_Insert+Insert, HDE_VNS; - Outras Características: Regra LOV.
Qian et al. (2009b)	- Função Objetivo: Bicritério; - Critério de Desempenho: *Makespan* e Atraso Máximo; - Restrições Adicionais: *Buffers* limitados.	- Estratégia de DE utilizada: *DE/target-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 50; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: 300; - Número Máximo de Runs: 10; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Carlier, Reeves e Taillard; - Métodos Utilizados para Comparação: IMMGLS2; - Outras Características: Regra LOV.

Qian et al. (2009c)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *Buffers* limitados.	- Estratégia de DE utilizada: *DE/target-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: *n*m*; - Número Máximo de Runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Carlier, Reeves e Taillard; - Métodos Utilizados para Comparação: HDE_NOL, HDE_ML, HDE_BL e OSA; - Outras Características: Regra LOV.
Tasgetiren et al. (2010)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flowtime*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: 10; - Taxa de Mutação (F): 0,9; - Taxa de Cruzamento (CR): 0,2; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional (T_{max} = 0,4 *nm* s); - Número Máximo de Gerações: não especifica; - Número de Runs: 10; - Hibridizações: IG e VNS; - *Benchmark* Utilizado: Jarboui, Eddaly e Siarry (2009) e Tseng e Lin (2009); - Métodos Utilizados para Comparação: hGLS, VNS, EDA e DBAC ; - Outras Características: ----.

Xu, Xiang e Wang (2010)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 5*n*; - Taxa de Mutação (F): 0,8; - Taxa de Cruzamento (CR): 0,5; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Maximo de Gerações: 400; - Número de Runs: 30; - Hibridizações ---- ; - *Benchmark* Utilizado: Carlier e Reeves; - Métodos Utilizados para Comparação: DE e HDE_NOL; - Outras Características: A Teoria do Caos é utilizada para otimizar os parâmetros; Regra LOV.
Zheng e Yamashiro (2010)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; *Flowtime*; e *Lateness*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/2/exp*; - Inicialização da População: A DE é aplicada a cada Q-bit para todos os crossomoso quantum gerados na fase de inicialização do algoritmo QEA (Quantum Evolutionary Algorithm); - Tamanho da População: 50; - Taxa de Mutação (F): 0,9; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações ou obtenção do ótimo; - Número Máximo de Gerações: 500; - Número de Runs: 20; - Hibridizações: Algoritmo Evolucionário Quantum-Inspirado e VNS; - *Benchmark* Utilizado: Carlier, Heller, Reeves e Taillard; - Métodos Utilizados para Comparação: PSO, DPSO, HQEA, HGA, HDE, BEST(LR), M-MMAS, ACO, PACO, LS e PSO-LS; - Outras Características: Regra LRVA para determinar a sequencia de tarefas no cromossomo quantum.

Mokhtari, Abadi e Cheraghalikhami (2011)	- Função Objetivo: Bicritério; - Critério de Desempenho: *Makespan* e Custo dos Recursos - Restrições Adicionais: Tempos de processamento dependente dos recursos (RDPT)	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Regras SPT, LPT, SOT, LOT e Aleatória; - Tamanho da População: {20, 40, 50, 100, 300}; - Taxa de Mutação (F): {0,7, 0,9}; - Taxa de Cruzamento (CR): {0,05, 0,2}; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Nùmero máximo de gerações; - Número Máximo de Gerações: {50, 80, 100, 150, 200, 350}; - Número Máximo de Runs: 10; - Hibridizações: VNS; - *Benchmark* Utilizado: Reeves, Carlier e Taillard; - Métodos Utilizados para Comparação: ODE, PGA, NEH, Palmer, CDS e HDDE; - Outras Características: Regra LOV.
Tasgetien et al. (2011)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Tardiness*; - Restrições Adicionais: *No-Idle*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): {0,1, 0,9}; - Taxa de Cruzamento (CR): {0,1, 0,9}; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional (T_{max} = 10*n* ms); - Número Máximo de Gerações----- ; - Número Máximo de Runs----- ; - Hibridizações: VPS; - *Benchmark* Utilizado: Taillard e as datas de entrega (*due dates*) são determinadas conforme Ruiz e Stutzle (2006); - Métodos Utilizados para Comparação: RKGA; - Outras Características: Regra LPV.

Deng e Gu (2012)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Idle*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH, variante da *Insertion Improvement* e aleatória; - Tamanho da População: 20; - Taxa de Mutação (F): 0,8; - Taxa de Cruzamento (CR): 0,8; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Máximo Tempo Computacional (T_{max} = nm/2tp ms, com tp = {20, 40, 60}); - Número Máximo de Gerações----- ; - Número Máximo de Runs: 15000; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: NEHD, $NEHD_{nsu}$, IG_{LS}, HDPSO e DDE_{LS}; - Outras Características: Método de aceleração baseado em representação de redes.
Hu et al. (2012)	- Função Objetivo: Monocritério; - Critério de Desempenho: Número de Tarefas em atraso - Restrições Adicionais: *No-Wait*; *Setup*; *Release Dates*.	- Estratégia de DE utilizada: *DE/rand-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Máximo tempo computacional (T_{max} = *2nm* ms); - Número máximo de gerações ---- ; - Número máximo de runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: IG, HDE, DE_NTJ_V1 e DE_NTJ_V2; - Outras Características: Regra LOV; Método de aceleração.

Liu (2012)	- Função Objetivo: Moncoritério; - Critério de Desempenho: *Flowtime*; - Restrições Adicionais: Não Há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: 100; - Taxa de Mutação (F): 0,5; - Taxa de Cruzamento (CR): Aleatória baseado em distribuição Gaussiana com média 0.5 e desvio padrão 0.1 e então truncado em [0,1] ; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Máximo Tempo Computacional (Tmax = *nm*0,1 s); - Número Máximo de Gerações----- ; - Número Máximo de Runs----- ; - Hibridizações: SA e VNS; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: PSO_{VNS} e H-CPSO; - Outras Características: Regra SPV.
Lobato, Gedraite e Neiro (2012)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Horizonte de Tempo e tamanhos mínimo e máximo de lotes.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 500; - Taxa de Mutação (F): {0,1, 0,5, 0,8, 1,0, 2,0}; - Taxa de Cruzamento (CR): {0,1, 0,3, 0,5, 0,8, 0,9}; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Número máximo de avaliações da função objetivo; - Número Máximo de Gerações: 10000; - Número Máximo de Runs: 20; - Hibridizações ---- ; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: Modelo Matemático implementado em GAMS; - Outras Características: ----.

Qian et al. (2012a)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*; *Setups*; e *Release Dates*.	- Estratégia de DE utilizada: *DE/rand-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações e Tempo Máximo Computacional (tempo máximo computacional igual ao do algoritmo RT); - Número Máximo de Gerações: $40n^2$ para o algoritmo SAFM; - Número Máximo de Runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: HDE, SAFM e IG; - Outras Características: Regra LOV; Método de aceleração.
Qian et al. (2012b)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*; *Setups*; e *Release Dates*.	- Estratégia de DE utilizada: *DE/rand-to-best/1/exp*; - Inicialização da População: Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações e Tempo máximo computacional (Tmax igual ao do algoritmo utilizado para comparação); - Número Máximo de Gerações: $40n$ para o SAFM; - Número Máximo de Runs: 30; - Hibridizações: Busca local; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: SAFM, HDE_FSSP, HDE_FSSP_SP e IG; - Outras Características: Regra LOV; Método de aceleração.

Tasgetiren et al. (2012a)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flowtime*; - Restrições Adicionais: não há.	- Estratégia de DE utilizada: *DE /rand/*1*/ bin*; - Inicialização da População: heurística NEH e aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,9; - Taxa de Cruzamento (CR): 0,9; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional (T_{max} = 10*n* ms); - Número Maximo de Gerações-----; - Número máximo de Runs: 5; - Hibridizações: VIG e busca local; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: vIG_FR; - Outras Características: ----.
Tasgetiren et al. (2012b)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Idle*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: heurística NEH e Aleatória; - Tamanho da População: 30; - Taxa de Mutação (F): 0,9; - Taxa de Cruzamento (CR): 0,9; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional (T_{max} = *n(m/2)t* ms com t=30); - Número Máximo de Gerações-----; - Número máximo de Runs: 5; - Hibridizações: IGA; - *Benchmark* Utilizado: Ruiz; - Métodos Utilizados para Comparação: IG_LS, HDDE; - Outras Características: ----.

Akrout et al. (2013a)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 50; - Taxa de Mutação (F): 1,5; - Taxa de Cruzamento (CR): 0,85; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional ($n^2/2$ milissegundos); - Número Máximo de Gerações-----; - Número Máximo de Runs-----; - Hibridizações: GRASP e ILS; - *Benchmark* Utilizado: Reeves, Heller e Taillard; - Métodos Utilizados para Comparação: RAJ, VNS_{SF}, $GASA_{SF}$, DS_{GP}, $DS+M_{GP}$, TS_{GP}, $TS+M_{GP}$, $TS+MP_{GP}$, HGA_{TL}, $DPSO_{PTL}$, $DPSOVND_{PTL}$ e GA-VNS_{JES}; - Outras Características-----;
Akrout et al. (2013b)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 50; - Taxa de Mutação (F): 1,5; - Taxa de Cruzamento (CR): 0,85; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações-----; - Número de Runs: 10; - Número Máximo de Avaliações da Função-Objetivo ----; - Hibridizações: GRASP e ILS; - *Benchmark* Utilizado: Taillard, Reeves e Heller; - Métodos Utilizados para Comparação: RAJ, VNS_{SF}, $GASA_{SF}$,DS_{GP}, $DS+M_{GP}$, TS_{GP}, $TS+M_{GP}$, $TS+MP_{GP}$, HGA_{TL}, $DPSO_{PTL}$, $DPSOVND_{PTL}$ e GA-VNS_{JES}; - Outras Características-----;

Guimarães et al. (2013)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin* e *DE/best/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 10; - Taxa de Mutação (F): {1, 0,8, 1}; - Taxa de Cruzamento (CR): {0,2, 0,9, 1}; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo Máximo computacional (T_{max} = $n(m/2)t$ ms, com t = 30 ms, t = 40 ms e t = 60 ms); - Número Máximo de Gerações----- ; - Número Máximo de Runs: 5; - Hibridizações: RLS; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: IG_RS, IG_RS$_{LS}$, NEHT, GA_RMA, AS_OP, SPIRIT, HGA_RMA, GA_CHEN, GA_REEV, GA_MIT, ILS, GA_AA, M-MMAS, PACO e DDERLS; - Outras Características: Regra LOV; Regra RIP; Transformação Forward Backward; Matriz de Adjacência; Matriz de Permutação.
Li e Yin (2013)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; e *Lateness*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/target-to-best/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: {60, 2*n*}; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): adaptativa; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: {300, 500, 1000}; - Número Máximo de Runs: 15; - Hibridizações: Algoritmo Memético, Busca Local e Busca Baseada em Oposição; - *Benchmark* Utilizado: Carlier, Reeves, Taillard e Demirkol et al. (1998); - Métodos Utilizados para Comparação: DE, HDE, OSA, PSOMA, PSOVNS, SGA, SGA+NEH, HGA, HQEA, HQDEA, ATPPSO, L-CDPSO e LWK_SA1; - Outras Características: Regra LRV.

Qian et al. (2013)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Diferentes tempos de reentrada das tarefas (Fluxo Reentrante)	- Estratégia de DE: *DE/rand-to-best/1/exp*; - Inicialização da População: Aletória (população do ED); NEH, NDYe SPT/TWKR (população do HGA); - Tamanho da População: 30; - Taxa de Mutação (F): 0,7; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério máximo de Parada: Número máximo de gerações; - Número Máximo de Gerações: 150; - Número Máximo de execuções/Runs: 20. - Hibridizações: Busca local; - *Benchmark* Utilizado: Gerado pelos autores; - Métodos Utilizados para Comparação: HGA, MRPFSSP e variantes e DE_TST; - Outras Características: Regra LOV.
Tasgetiren et al. (2013)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; e *Flowtime*; - Restrições Adicionais: *No-Idle*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH; - Tamanho da População: 30; - Taxa de Mutação (F): 0,9; - Taxa de Cruzamento (CR): 0,9; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo Máximo Computacional (Tmax = $n(m^2)t$ ms, com t=30 ms); - Número Máximo de Gerações----- ; - Número Máximo de Runs: 5; - Hibridizações: IG; - *Benchmark* Utilizado: Ruiz; - Métodos Utilizados para Comparação: HDDE, DDE_LS, HDPSO, IG_LSIG_RIS, VIG_FL; - Outras Características: Método de aceleração.

Tonge e Kulkarni (2013)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: {400, 700, 2500, 7000, 9000}; - Taxa de Mutação (F): {0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9}; - Taxa de Cruzamento (CR): {0,1, 0,3, 0,6, 0,7, 0,8}; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada -----; - Número máximo de gerações ---- ; - Número máximo de runs----- ; - Hibridizações: NEH e ILS; - *Benchmark* Utilizado: Carlier; - Métodos Utilizados para Comparação: GA e QIDE; - Outras Características: ----.
Amirian e Sahraeian (2014)	- Função Objetivo: Bicritério; - Critério de Desempenho: *Makespan* e *Flowtime*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/best/1/bin* e *DE/rand-to-best/1/exp*; - Inicialização da População: Busca Local; - Tamanho da População: 100; - Taxa de Mutação (F): {0,6023, 0,80233}; - Taxa de Cruzamento (CR): 0,2; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número máximo de gerações: 100; - Número máximo de runs: 10; - Hibridizações: Busca local e NSGA-II; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: PASA, MPFA, MOSA e PGA-ALS; - Outras Características: Regra LOV.

Davendra et al. (2014)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/best/1/exp*; - Inicialização da População: Conforme Davendra e Onwubolu (2009); - Tamanho da População: 100; - Taxa de Mutação (F): Conforme Davendra e Onwubolu (2009); - Taxa de Cruzamento (CR): Conforme Davendra e Onwubolu (2009); - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: 100; - Número Máximo de Runs----- ; - Hibridizações ---- ; - *Benchmark* Utilizado -----; - Métodos Utilizados para Comparação----- ; - Outras Características: Transformação *Foward e Bacward*; Os atributos da dinâmica populacional do algoritmo de Evolução Diferencial são analidados usando ferramentas de análise de redes complexas; A população é visualizado como uma rede complexa em evolução.
Liu, Yin e Gu (2014)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística NEH e Aleatória; - Tamanho da População: {60, 2n}. - Taxa de Mutação (F): 2; - Taxa de Cruzamento (CR): 0,4; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações: {500, 1000}; - Número Máximo de Runs----- ; - Hibridizações: IIS-Based Local Search e Greedy- Based Local Search; - *Benchmark* Utilizado: Taillard, Carlier e Reeves; - Métodos Utilizados para Comparação: Variantes da DE (DE, HDE e L-HDE), ATPPSO-R, ATPPSO, HPSO, NPSO, PSOMA, PSOVNS, SGA, SGA+NEH e HGA; - Outras Características: Regra LRV.

Santuci, Baioletti e Milani (2014)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flowtime*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Heurística *LR*(*n/m*) e Aleatória; - Tamanho da População: 100; - Taxa de Mutação (F): Autoadaptativo; - Taxa de Cruzamento (CR)----- ; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada; ----; - Número máximo de gerações ---- ; - Número máximo de runs: 20; - Hibridizações: Busca Local; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: AGA, VNS_4, GM_EDA e HGM_EDA; - Outras Características: o cruzamento é realizado de acordo com a versão II do cruzamento de dois pontos (TPII) proposto por Murata, Ishibuchi e Tanaka (1996) e usado no AGA de Xu, Xu e Gu (2011); Algoritmo *Bubble Sort* (*RandBS*).
Dong (2015)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *No-Wait*.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: Entre 30 e *n*; - Taxa de Mutação (F): 0,5; - Taxa de Cruzamento (CR): 0,5; - Procedimento de Seleção: individuo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo de execução (2n(s), com n < 30 e 5n(s) se n > 30); - Número Máximo de Gerações----- ; - Número de runs: 20; - Hibridizações: TS e Busca Local (FCH); - *Benchmark* Utilizado: Reeves e Heller; - Métodos Utilizados para Comparação: RAJ, HDE e $DPSO_{VSD}$; - Outras Características: O conceito de TS é implementado no passo de geração do DE (antes da mutação e o cruzamento); A Busca Local pode ou não ser usada na após a mutação e o cruzamento.

Tasgetiren et al. (2015)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: *Blocking*.	- Estratégia de DE utilizada: *DE/best/1/bin*; - Inicialização da População: Heurística PF_NEH e GRASP_NEH; - Tamanho da População: 10; - Taxa de Mutação (F): 0,1; - Taxa de Cruzamento (CR): 0,1; - Procedimento de Seleção: Índividuo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo de execução (100*nm* milisegundos); - Número Máximo de Gerações: 20; - Número de runs: 5; - Hibridizações: ILS e IG; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: HDDE, RAIS, IG, IG-reimplementada e MA; - Outras Características-----;
Tien et al. (2015)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/best/1/bin*; - Inicialização da População: Aleatória; - Tamanho da População: 100 - Taxa de Mutação (F): {0,4, 0,8}; - Taxa de Cruzamento (CR): não espeficifica o valor (configurada no limiar); - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Número máximo de gerações; - Número Máximo de Gerações-----; - Número de runs: 30; - Hibridizações ----; - *Benchmark* Utilizado: Taillard (8 problemas testes); - Métodos Utilizados para Comparação: EDA; - Outras Características: Um novo mecanismo de mutação é usado para permitir um sequenciamento apropriado para cada tarefa.

Santucci, Baioletti e Milani (2016)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Flowtime*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*;; - Inicialização da População: Heurísticas R_INIT e H_INIT ; - Tamanho da População: {50, 100, 200}; - Taxa de Mutação (F): autoadaptativo; - Taxa de Cruzamento (CR)----- ; - Procedimento de Seleção: esquema de seleção tendencioso, com parâmetro de ajuste θ . $\theta = \{0, 0{,}005, 0{,}010, 0{,}2\}$ - Critério de Parada: Número máximo de avaliações da função objetivo por geração, conforme Pan e Ruiz (2012); - Número Máximo de Gerações----- ; - Número Máximo de Runs: 10; - Hibridizações: Busca local; - *Benchmark* Utilizado: Taillard; - Métodos Utilizados para Comparação: AGA, VNS4, GM-EDA, HGM-EDA, ILS e IGA; - Outras Características: Mutação diferencial algébrica; Operador de cruzamento para permutações; procedimento memético de reinício.
Shao e Pi (2016)	- Função Objetivo: Monocritério; - Critério de Desempenho: *Makespan*; - Restrições Adicionais: Não há.	- Estratégia de DE utilizada: *DE/rand/1/bin*; - Inicialização da População: Algoritmos DHS, NEH, RAJ FBR1 e Aleatória; - Tamanho da População: 100; - Taxa de Mutação (F): 0,4; - Taxa de Cruzamento (CR): 0,3; - Procedimento de Seleção: Indivíduo com menor valor da função-objetivo; - Critério de Parada: Tempo máximo computacional (T_{max} = *nm*60 ms e T_{max} = *nm*10 ms); - Número Máximo de Gerações----- ; - Número Máximo de Runs----- ; - Hibridizações: VNS (INSERT_VNS e SWAP_VNS); - *Benchmark* Utilizado: Carlier, Reeves e Taillard; - Métodos Utilizados para Comparação: SGDE_NOHS, SGDE, EDA, DE, PSOMA, HBSA, HTLBO, ATPPSO, ODDE e ACGA; - Outras Características: ----.

APÊNDICE B – Regra LOV: Exemplo Ilustrativo

Dado um problema com 6 tarefas (n=6) onde o indivíduo X_i é X_i=[1,36,3,85,2,55,0,63,2,68,0,82].

De acordo com a regra LOV, pelo fato de $x_{i,2}$ ser o maior valor de X_i, $x_{i,2}$ é selecionado primeiro e designado a posição 1 da sequencia. Então $x_{i,5}$ é o segundo selecioando e designado a posição 2 da sequencia. Da mesma forma $x_{i,3}$, $x_{i,1}$, $x_{i,6}$ e $x_{i,4}$ são designados as posições 3, 4, 5 e 6, respectivamente. Deste modo a sequencia φ_i=[4, 1, 3, 6, 2, 5].

De acordo com a equação (5.2) apresentada no capítulo 5, se k = 1, então $\varphi_{i,1}=4$ e $\pi_{i,\varphi_{i,1}}=\pi_{i,4}=1$; se k = 2, então $\varphi_{i,2}=1$ e $\pi_{i,\varphi_{i,2}}=\pi_{i,1}=2$; se k = 3, então $\varphi_{i,3}=3$ e $\pi_{i,\varphi_{i,3}}=\pi_{i,3}=3$; se k = 4, então $\varphi_{i,4}=6$ e $\pi_{i,\varphi_{i,4}}=\pi_{i,6}=4$; se k = 5, então $\varphi_{i,5}=2$ e $\pi_{i,\varphi_{i,5}}=\pi_{i,2}=5$; se k = 6, então $\varphi_{i,6}=5$ e $\pi_{i,\varphi_{i,6}}=\pi_{i,5}=6$. Deste modo tem-se a permutação de tarefas π_i=[2,5,3,1,6,4].

O processo de represenção da solução por meio da regra LOV, anteriormente descrito, é ilustrado na Tabela B.1 a seguir.

Tabela B.1 - Representação da Solução: Regra LOV

Dimensão k	1	2	3	4	5	6
$x_{i,k}$	1,36	3,85	2,55	0,63	2,68	0,82
$\varphi_{i,k}$	4	1	3	6	2	5
$\pi_{i,k}$	2	5	3	1	6	4

Fonte: Qian et al., 2008.

APÊNDICE C – Descrição do *Benchmark* de Carlier

Caso 1 - Car_001_11_05					
Número de Tarefas		Número de Máquinas		C_{max} (Limitante Superior)	
11		5		7038	
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina					
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5
j_1	375	12	142	245	412
j_2	632	452	758	278	398
j_3	12	876	124	534	765
j_4	460	542	523	120	499
j_5	528	101	789	124	999
j_6	796	245	632	375	123
j_7	532	230	543	896	452
j_8	14	124	214	543	785
j_9	257	527	753	210	463
j_{10}	896	896	214	258	259
j_{11}	532	302	501	765	988

Caso 2 - Car_002_13_04				
Número de Tarefas	Número de Máquinas		C_{max} (Limitante Superior)	
13	4		7166	
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina				
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4
j_1	654	147	345	447
j_2	321	520	789	702
j_3	12	147	630	255
j_4	345	586	214	866
j_5	678	532	275	332
j_6	963	145	302	225
j_7	25	24	142	589
j_8	874	517	24	996
j_9	114	896	520	541
j_{10}	785	543	336	234
j_{11}	203	210	699	784
j_{12}	696	784	855	512
j_{13}	302	512	221	345

Caso 3 - Car_003_12_05					
Número de Tarefas		Número de Máquinas		C_{max} (Limitante Superior)	
12		5		7312	
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina					
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5
j_1	456	537	123	214	234
j_2	789	854	225	528	123
j_3	876	632	588	896	456
j_4	543	145	669	325	789
j_5	210	785	966	147	876
j_6	123	214	332	856	543
j_7	456	752	144	321	210
j_8	789	143	755	427	123
j_9	876	698	322	546	456
j_{10}	543	532	100	321	789
j_{11}	210	145	114	401	876
j_{12}	124	247	753	214	543

Caso 4 - Car_004_14_04				
Número de Tarefas	Número de Máquinas		C_{max} (Limitante Superior)	
14	4		8003	
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina				
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4
j_1	456	856	963	696
j_2	789	930	21	320
j_3	630	214	475	142
j_4	214	257	320	753
j_5	573	896	124	214
j_6	218	532	752	528
j_7	653	142	147	653
j_8	214	547	532	214
j_9	204	865	145	527
j_{10}	785	321	763	536
j_{11}	696	124	214	214
j_{12}	532	12	257	528
j_{13}	12	345	854	888
j_{14}	457	678	123	999

Caso 5 - Car_005_10_06						
Número de Tarefas		Número de Máquinas		C_{max} (Limitante Superior)		
10		6		7720		
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina						
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5	i_6
j_1	333	991	996	123	145	234
j_2	333	111	663	456	785	532
j_3	252	222	222	789	214	586
j_4	222	204	114	876	752	532
j_5	255	477	123	543	143	142
j_6	555	566	456	210	698	573
j_7	558	899	789	124	532	12
j_8	888	965	876	537	145	14
j_9	889	588	543	854	247	527
j_{10}	999	889	210	632	451	856

Caso 6 - Car_006_08_09									
Número de Tarefas			Número de Máquinas			C_{max} (Limitante Superior)			
8			9			8505			
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina									
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
j_1	887	447	234	159	201	555	463	456	753
j_2	799	779	567	267	478	444	123	789	21
j_3	999	999	852	483	520	120	456	630	427
j_4	666	666	140	753	145	142	789	258	520
j_5	663	25	222	420	699	578	876	741	142
j_6	333	558	558	159	875	965	543	36	534
j_7	222	886	965	25	633	412	210	985	157
j_8	114	541	412	863	222	25	123	214	896

Caso 7 - Car_007_07_07							
Número de Tarefas		Número de Máquinas			C_{max} (Limitante Superior)		
7		7			6590		
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina							
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5	i_6	i_7
j_1	692	310	832	630	258	147	255
j_2	581	582	14	214	147	752	806
j_3	475	475	785	578	852	2	699
j_4	23	196	696	214	586	356	877
j_5	158	325	530	785	325	565	412
j_6	796	874	214	236	896	898	302
j_7	542	205	578	963	325	800	120

Caso 8 - Car_008_08_08								
Número de Tarefas		Número de Máquinas			C_{max} (Limitante Superior)			
8		8			8366			
Tempo de processamento (t_{ij}) da *j*-ésima tarefa na *i*-ésima máquina								
Tarefa (*j*)/Máquina(*i*)	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
j_1	456	654	852	145	632	425	214	654
j_2	789	123	369	678	581	396	123	789
j_3	654	123	632	965	475	325	456	654
j_4	321	456	581	421	32	147	789	123
j_5	456	789	472	365	536	852	654	123
j_6	789	654	586	824	352	12	321	456
j_7	654	321	320	758	863	452	456	789
j_8	789	147	120	639	21	863	789	654