UNIVERSIDADE FEDERAL DO PARANÁ

FLÁVIO AUGUSTO FERREIRA DO NASCIMENTO

PLANEJAMENTO FLORESTAL COM RESTRIÇÕES DE ADJACÊNCIA UTILIZANDO PROGRAMAÇÃO PARALELA

CURITIBA

2014

FLÁVIO AUGUSTO FERREIRA DO NASCIMENTO

PLANEJAMENTO FLORESTAL COM RESTRIÇÕES DE ADJACÊNCIA UTILIZANDO PROGRAMAÇÃO PARALELA

Tese apresentada como requisito parcial à obtenção do grau de Doutor, pelo Programa de Pós Graduação em Engenharia Florestal – Área Manejo Florestal, do Setor de Ciências Agrárias da Universidade Federal do Paraná.

Orientador: Prof. Dr. Julio Eduardo Arce
Co-Orientador: Prof Dr. Afonso Figueiredo Filho
Co-Orientador: Prof. Dr. Arinei Carlos Lindbeck da Silva

CURITIBA
2014

Biblioteca de Ciências Florestais e da Madeira - UFPR
Ficha catalográfica elaborada por Denis Uezu – CRB 1720/PR

Nascimento, Flávio Augusto Ferreira do

Planejamento florestal com restrições de adjacência utilizando programação paralela / Flávio Augusto Ferreira do Nascimento. – 2014

130 f. : il.

Orientador: Prof. Dr. Julio Eduardo Arce

Coorientador: Prof Dr. Afonso Figueiredo Filho

Prof. Dr. Arinei Carlos Lindbeck da Silva

Tese (doutorado) - Universidade Federal do Paraná, Setor de Ciências Agrárias, Programa de Pós-Graduação em Engenharia Florestal. Defesa: Curitiba, 27/08/2014.

Área de concentração: Manejo Florestal

1. Manejo florestal. 2. Colheita florestal. 3. Pesquisa operacional – Processamento de dados. 4. Otimização matemática. 5. Teses. I. Arce, Julio Eduardo. II. Figueiredo Filho, Afonso. III. Silva, Arinei Carlos Lindbeck da. IV. Universidade Federal do Paraná, Setor de Ciências Agrárias. V. Título.

CDD – 634.9

CDU – 634.0.31

Universidade Federal do Paraná
Setor de Ciências Agrárias - Centro de Ciências Florestais e da Madeira
Programa de Pós-Graduação em Engenharia Florestal

PARECER

Defesa nº. 1056

A banca examinadora, instituída pelo colegiado do Programa de Pós-Graduação em Engenharia Florestal, do Setor de Ciências Agrárias, da Universidade Federal do Paraná, após argüir o(a) doutorando(a) *Flavio Augusto Ferreira do Nascimento* em relação ao seu trabalho de tese intitulado "***PLANEJAMENTO FLORESTAL COM RESTRIÇÕES DE ADJACÊNCIA UTILIZANDO PROGRAMAÇÃO PARALELA***", é de parecer favorável à *APROVAÇÃO* do(a) acadêmico(a), habilitando-o(a) ao título de *Doutor* em Engenharia Florestal, área de concentração em MANEJO FLORESTAL.

Dr. Volmir Eugênio Wilhelm
Univeridade Federal do Paraná
Primeiro examinador

Dr. Lucas Rezende Gomide
Universidade Federal de Lavras
Segundo examinador

Dr. Andrea Nogueira Dias
Universidade Estadual do Centro-Oeste
Terceiro examinador

Dr. Arinei Carlos Lindbeck da Silva
Universidade Federal do Paraná
Quarto examinador

Dr. Julio Eduardo Arce
Universidade Federal do Paraná
Orientador e presidente da banca examinadora

MINISTÉRIO DA EDUCAÇÃO - U.F.P.R. - Curso de Pós Graduação em Engenharia Florestal - MESTRADO DOUTORADO - Setor de Ciências Agrárias

Curitiba, 27 de agosto de 2014.

Antonio Carlos Batista
Coordenador do Curso de Pós-Graduação em Engenharia Florestal

AGRADECIMENTOS

A minha família pelo apoio incondicional durante toda a minha vida. Aos meus pais Swami e Sirlei, aos meus irmãos Marco, Giovana (*in memoriam*) e Julio, a minha cunhada Manoelita e, principalmente a minha esposa Regina, que esteve do meu lado durante grandes dificuldades e que diariamente consegue ter a paciência de aturar as minhas piadas ruins, meus trocadilhos infames e minhas teorias baseadas em referências baratas.

Ao professor Dr. Julio Eduardo Arce, pela orientação objetiva e eficaz, bem como pela amizade e companheirismo, mas em especial por servir de inspiração para eu me tornar um profissional/pesquisador/pessoa melhor.

Ao professor Dr. Afonso Figueiredo Filho pelo apoio e orientações e por ter me ensinado todos os segredos do manejo florestal. Ao professor Dr. Arinei Carlos Lindbeck da Silva por ter me mostrado que a programação computacional não precisa ser necessariamente indistinguível de magia. A professora Dr. Andrea Nogueira Dias por ter me iniciado ao mundo da pesquisa operacional e ter me apoiado durante todas as etapas da minha formação acadêmica (graduação, mestrado e doutorado). Aos demais membros da banca Dr. Lucas Resende Gomide e Dr. Volmir Eugênio Wilhelm pelas contribuições e melhorias à tese.

Aos amigos, colegas de doutorado e colegas de trabalho que, de alguma forma, direta ou indiretamente, contribuíram para o desenvolvimento deste trabalho.

RESUMO

O presente trabalho teve como objetivo avaliar versões paralelas e sequenciais dos algoritmos enxame de partículas (*particle swarm optimization* – PSO) e busca tabu (BT), aplicados a modelos de planejamento florestal com diferentes tipos de restrições de adjacência entre talhões. Os dados foram obtidos de povoamentos de *Pinus* spp e *Eucalyptus* spp com área total de 2.429,99 ha e 236 talhões, pertencentes à empresa REMASA Reflorestadora S.A. Foram gerados cenários considerando os tipos de restrição de adjacência denominados *unit restriction model* (URM) e *area restricition model* (ARM). Os métodos de resolução dos cenários foram os algoritmos PSO, BT e o método exato *branch and bound*. Foi também proposta uma modificação ao algoritmo PSO com intensificação de busca em vizinhança. Para os três métodos, PSO, BT e modificação da PSO, foram avaliadas diferentes configurações de seus parâmetros. Como forma de melhorar o tempo de processamento das meta-heurísticas, foram implementadas versões paralelas dos algoritmos PSO e BT conforme o modelo *fork-and-join* empregando a biblioteca OpenMP. Os resultados demonstram que o modelo ARM gera maior quantidade de restrições que o modelo URM. As melhores configurações para o algoritmo PSO foram com velocidade máxima de 10% e populações maiores que 100 partículas. Para a modificação proposta ao algoritmo PSO a melhor configuração foi com população de 10 partículas e intensificação de busca em vizinhança de uma solução vizinha por partícula por iteração. O algoritmo BT apresenta as melhores soluções quando a busca é realizada na vizinhança total e o tempo tabu é igual a 10 iterações. Quando comparado o desempenho dos três algoritmos, a BT apresenta as soluções de melhor qualidade, seguido da modificação da PSO com busca em vizinhança e, por fim, o algoritmo original PSO. As versões paralelas dos algoritmos PSO e BT apresentaram economia de tempo de processamento em relação às versões sequenciais, indicando que a programação paralela é uma estratégia promissora para resolução de problemas de planejamento florestal de grande porte.

Palavras-chave: restrições espaciais, meta-heurísticas, agendamento da colheita florestal.

ABSTRACT

This study aimed to evaluate parallel and sequential versions of particle swarm optimization (PSO) and tabu search (BT) algorithms, applied to forest planning models with different types of adjacent constraints between stands. Data were obtained from stands of *Pinus* spp and *Eucalyptus* spp with total area of 2,429.99 ha and 236 stands belonging to the REMASA Reflorestadora S.A. enterprise. Scenarios were generated considering the adjacency constraint denominated unit restriction model (URM) and area restricition model (ARM). The methods of solving scenarios were the PSO, BT algorithms and the exact method branch and bound. It has also proposed a modification to the PSO algorithm by intensifying the search in the neighborhood. For the three methods, PSO, BT and modification of PSO have been evaluated different settings of its parameters. In order to improve the processing time of the meta-heuristics, parallel versions of PSO and BT algorithms have been implemented as the fork-and-join model using the OpenMP library. The results demonstrate that although less restrictive, the ARM model generates more constraints that the URM. The best settings for the PSO algorithm have been maximum speed of 10% and populations greater than 100 particles. To the proposed modification to the PSO algorithm was the best configuration with a population of 10 particles and intensification search in a neighboring solution per particle per iteration neighborhood. The BT algorithm presents the best solutions when the search is performed in the full neighborhood and tabu time is equal to 10 iterations. When comparing the performance of three algorithms, BT has the better solutions, followed by modification of the PSO using neighborhood search, and finally, the original PSO algorithm. The parallel versions of PSO and BT algorithms showed processing time savings compared to the sequential versions, indicating that parallel programming is a promising strategy for solving problems of large forest planning.

Key-words: spatial constraints, meta-heuristics, harvesting scheduling.

LISTA DE TABELAS

TABELA 1 - ESPÉCIES MANEJADAS NA ÁREA DE ESTUDO. 53
TABELA 2 - CLASSES DE SÍTIO PARA O *Pinus taeda* NA ÁREA DE ESTUDO. 54
TABELA 3 - PREÇOS DE VENDA DE MADEIRA PARA ÁRVORES EM PÉ... 58
TABELA 4 - CUSTOS DE PRODUÇÃO PARA *Pinus* spp..... 58
TABELA 5 - DIÂMETRO DAS TORAS DE CADA PRODUTO CONSIDERADO. 60
TABELA 6 - CENÁRIOS DE PLANEJAMENTO FLORESTAL EMPREGADOS NOS TESTES..... 63
TABELA 7 - CONFIGURAÇÃO DE PARÂMETROS AVALIADOS PARA A PSO. 77
TABELA 8 - CONFIGURAÇÃO DE PARÂMETROS AVALIADOS PARA A BT 78
TABELA 9 - DIMENSÕES DOS CENÁRIOS EMPREGADOS NOS TESTES . 81
TABELA 10 - RESULTADOS DO LINGO..... 82
TABELA 11 - RESULTADOS SU-SPSO PROBLEMA URM_HP20. 84
TABELA 12 - RESULTADOS NS-SPSO – PROBLEMA URM_HP20. 88
TABELA 13 - RESULTADOS DO ALGORITMO BT PARA O PROBLEMA URM_HP20 90
TABELA 14 - RESULTADOS DAS META-HEURÍSTICAS - PROBLEMA URM_HP20. 92
TABELA 15 - RESULTADO DO TESTE DE DUNN PARA O PROBLEMA URM_HP20. 95
TABELA 16 - RESULTADOS DAS META-HEURÍSTICAS - PROBLEMA ARM_HP20. 96
TABELA 17 - RESULTADO DO TESTE DE DUNN PARA O PROBLEMA ARM_HP20. 98
TABELA 18 - TEMPO MÉDIO DE PROCESSAMENTO DA SU-PPSO. 99
TABELA 19 - TEMPO MÉDIO DE PROCESSAMENTO DA BT PARALELA. 101

LISTA DE FIGURAS

FIGURA 1 - COMPORTAMENTO DAS PARTÍCULAS EM ESPAÇO BIDIMENSIONAL (ADAPTADO DE MÜLLER, 2007). 31
FIGURA 2 - TIPOS DE ARQUITETURAS DE COMPUTAÇÃO PARALELA (ADAPTADO DE CHANDRA *et al.*, 2001). 44
FIGURA 3 - MODELO DE PROGRAMAÇÃO *FORK-AND-JOIN* SUPORTADA NA OPENMP (ADAPTADO DE CHAPMAN *et al.*, 2008). 45
FIGURA 4 - MAPA COM A LOCALIZAÇÃO DOS TALHÕES UTILIZADOS 52
FIGURA 5 - DISTRIBUIÇÃO DE IDADE DOS POVOAMENTOS. 53
FIGURA 6 - VPL EM FUNÇÃO DA IDADE EM QUE SE REALIZA A ANÁLISE ECONÔMICA. 57
FIGURA 7 - EXEMPLO DE FLUXO DE CAIXA DE UMA ALTERNATIVA DE MANEJO. 59
FIGURA 8 - REPRESENTAÇÃO DE ADJACÊNCIA ENTRE TALHÕES. 61
FIGURA 9 - EXEMPLO DE CODIFICAÇÃO DE UMA SOLUÇÃO. 64
FIGURA 10 - MOVIMENTAÇÃO DE UMA PARTÍCULA EM UM CENÁRIO COM DOIS TALHÕES. 65
FIGURA 11 - FLUXOGRAMA DO ALGORITMO SU-SPSO. 66
FIGURA 12 - FLUXOGRAMA DO ALGORITMO IU-SPSO. 67
FIGURA 13 - REPRESENTAÇÃO DA BUSCA EM VIZINHANÇA DE UMA PARTÍCULA. 69
FIGURA 14 - FLUXOGRAMA DO ALGORITMO SU-PPSO. 71
FIGURA 15 - FLUXOGRAMA DO ALGORITMO BT. 73
FIGURA 16 - EXEMPLO DE GERAÇÃO DE VIZINHANÇA TOTAL DO ALGORITMO BT. 74
FIGURA 17 - FLUXOGRAMA DO ALGORITMO BT PARALELO. 75
FIGURA 18 - PRODUÇÃO PARA A SOLUÇÃO LINGO DO PROBLEMA URM_HP20. 83
FIGURA 19 - PRODUÇÃO PARA A SOLUÇÃO LINGO DO PROBLEMA ARM_HP20. 83

FIGURA 20 - VALORES MÉDIOS DAS SOLUÇÕES PARA A ABORDAGEM SU-SPSO PARA O PROBLEMA URM_HP20. 85
FIGURA 21 - VALORES MÉDIOS DAS SOLUÇÕES PARA A ABORDAGEM IU-SPSO PARA O PROBLEMA URM_HP20. 86
FIGURA 22 - COMPARAÇÃO DAS ABORDAGENS SU-SPSO E IU-SPSO (VALORES MÉDIOS, PROBLEMA URM_HP20, $Vmax$ = 10%)..... 87
FIGURA 23 - RESULTADOS DA NS-SPSO EM FUNÇÃO DO TAMANHO DA POPULAÇÃO. 88
FIGURA 24 - RESULTADOS DA NS-SPSO EM FUNÇÃO DO TAMANHO DA VIZINHANÇA..... 89
FIGURA 25 - DESEMPENHO MÉDIO DA BT - PROBLEMA URM_HP20. 91
FIGURA 26 - MELHORES SOLUÇÕES DA BT - PROBLEMA URM_HP20. ... 91
FIGURA 27 - VALORES DE EFICÁCIA DAS TRÊS MELHORES SOLUÇÕES DAS META-HEURÍSTICAS - PROBLEMA URM_HP20. 93
FIGURA 28 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ABORDAGEM IU-SPSO. 94
FIGURA 29 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ABORDAGEM N-SPSO. 94
FIGURA 30 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ALGORITMO BT. 95
FIGURA 31 - VALORES DE EFICÁCIA DAS TRÊS MELHORES SOLUÇÕES DAS META-HEURÍSTICAS - PROBLEMA ARM_HP20. 96
FIGURA 32 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ABORDAGEM IU-SPSO. 97
FIGURA 33 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ABORDAGEM N-SPSO. 97
FIGURA 34 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ALGORITMO BT. 98
FIGURA 35 - COMPARAÇÃO DA *SPEEDUP* PARALELA DA SU-SPSO. 100
FIGURA 36 - EFICIÊNCIA DE PARALELIZAÇÃO DA SU-PPSO. 100
FIGURA 37 - COMPARAÇÃO DA *SPEEDUP* DA BT PARALELA. 102
FIGURA 38 - EFICIÊNCIA DE PARALELIZAÇÃO DA BT PARALELA. 103

LISTA DE SÍMBOLOS

χ	coeficiente de constrição.
a_{ik}	matriz binária {0,1} de adjacência entre talhões, onde *i* = *k.*
c_1	coeficiente cognitivo.
c_2	coeficiente social.
C_{ij}	valor presente líquido de cada talhão *i*, manejado sob a alternativa *j.*
C_t	custo observado no período *t*, em R$.
E	eficiência de paralelização.
Ef	eficácia (%).
$f(x)$	valor da função-objetivo do problema.
f_0	valor da solução obtida pelo algoritmo exato.
$f_{meta-heurística}$	valor da melhor solução obtida em um conjunto de execuções de uma meta-heurística.
$f_p(x)$	valor da função-objetivo penalizada para a solução *x.*
$gbest_p$	melhor posição encontrada na vizinhança da partícula *p.*
G_i	conjunto de talhões adjacentes, ou grupos.
HP	quantidade de períodos do horizonte de planejamento, em anos.
it	iteração atual.
IT	número total de iterações do algoritmo; e it = iteração atual.
M	quantidade total de talhões.
N	quantidade total de alternativas de manejo *j* no talhão *i.*
n_A	número de observações do grupo *A.*
NA	número total de observações em todos os grupos ou amostras.
n_B	número de observações do grupo *B.*
n_{G_i}	quantidade de talhões adjacentes presentes no conjunto, ou grupo.
P	número de processadores.
p	partícula.
$pbest_p$	melhor posição que a partícula *p* já obteve durante a busca.
$Pmax_t$	produção máxima para o período *t*, em m³.

$Pmin_t$	produção mínima para o período *t*, em m³.
$\overline{\mathrm{R}}$	o ranking médio, ou seja, $\mathrm{RA} = \mathrm{RA}/\mathrm{nA}$.
r	taxa de juros.
R_A	soma dos rankings do grupo *A.*
$rand$	função aleatória de distribuição uniforme entre 0 e 1.
R_t	receita observada no período *t*, em R$.
S	*speedup.*
SE	erro padrão.
t	período do horizonte de planejamento, em anos.
t_1	tempo de execução do programa em um(1) processador.
t_p	tempo de execução do programa em *P* processadores.
V_{ijt}	volume (m³) do produto *p* produzido pelo *i*-ésimo talhão assinalado na *j*-ésima alternativa de manejo, no início do período *t*
V_{max}	velocidade máxima de uma partícula em uma iteração.
vp	penalização para cada unidade de produção violada.
v_p	velocidade da partícula *p.*
VPL	valor presente líquido do fluxo de caixa futuro de cada alternativa de manejo, em R$.
VPL_G	valor presente líquido global da floresta.
VT	violação total (m³) das restrições de produção (mínima e máxima) e de adjacência.
w	momento de inércia.
w_{fin}	momento de inércia final.
w_{ini}	momento de inércia inicial.
w_{it}	momento de inércia na iteração atual.
w_{tx}	taxa de variação do momento de inércia em cada iteração
X_{ij}	talhão *i* assinalado à alternativa *j.*
x_p	posição da partícula *p.*

SUMÁRIO

1. INTRODUÇÃO 15

2. OBJETIVOS GERAIS 18

2.1. Objetivos específicos 18

3. REVISÃO DE LITERATURA 19

3.1. Planejamento florestal 19

3.2. Pesquisa operacional 20

3.3. Modelos de planejamento florestal 22

3.3.1. Considerações espaciais no planejamento florestal 22

3.3.2. Avaliação econômica de florestas 26

3.4. Meta-heurísticas 28

3.4.1. Otimização por enxame de partículas (PSO) 28

3.4.1.1. Parâmetros da PSO 31

3.4.1.2. Modificações ao algoritmo PSO 33

3.4.1.3. Aplicações do algoritmo PSO no planejamento florestal 34

3.4.2. Busca tabu 35

3.4.2.1. Movimentos e vizinhança na busca tabu 36

3.4.2.2. Lista tabu e tempo tabu 38

3.4.2.3. Critério de aspiração 38

3.4.2.4. Memória de longo prazo 39

3.4.2.5. Modificações e melhorias ao algoritmo BT 39

3.4.2.6. Aplicações da busca tabu no planejamento florestal 40

3.4.3. Avaliação de meta-heurísticas 40

3.5. Computação paralela 42

3.5.1. API para desenvolvimento de aplicativos paralelos 43

3.5.2. Implementações paralelas da PSO 46

3.5.3. Implementações paralelas da BT 48
3.5.4. Avaliação de desempenho das implementações paralelas 50
3.6. *Softwares* de otimização matemática 50
4. MATERIAL E MÉTODOS 52
4.1. Dados utilizados 52
4.2. Geração das alternativas de manejo 54
4.3. Avaliação econômica das alternativas de manejo 55
4.4. Preço da madeira e custo de produção 57
4.5. Formulação dos modelos 59
4.6. Restrições de adjacência entre talhões 60
4.6.1. Restrição do tipo *unit restriction model* (URM) 61
4.6.2. Restrição do tipo *area restriction model* (ARM) 62
4.7. Cenários de planejamento florestal 62
4.8. Codificação das soluções 63
4.9. Algoritmo PSO sequencial 64
4.10. Proposta de modificação ao algoritmo PSO 68
4.11. Algoritmo PSO paralelo 70
4.12. Algoritmo BT sequencial 72
4.13. Algoritmo BT paralelo 74
4.14. Função de avaliação utilizada nas meta-heurísticas PSO e BT 75
4.15. *Software* LINGO 76
4.16. Parâmetros das meta-heurísticas 77
4.17. Avaliação das meta-heurísticas 78
4.18. Avaliação do desempenho dos algoritmos paralelos 80
4.19. Implementação das meta-heurísticas e computador utilizado 80
5. RESULTADOS E DISCUSSÃO 81
5.1. Diferenças entre os cenários URM e ARM e resultados LINGO 81

5.2. Escolha dos parâmetros do algoritmo SU-SPSO 83

5.3. Escolha dos parâmetros do algoritmo IU-SPSO 85

5.4. Comparação dos algoritmos SU-SPSO e IU-SPSO 86

5.5. Escolha dos parâmetros do algoritmo NS-SPSO 87

5.6. Escolha dos parâmetros do algoritmo BT 89

5.7. Comparação das meta-heurísticas 92

5.8. Avaliação da versão paralela da PSO 99

5.9. Avaliação da versão paralela da BT 101

6. CONCLUSÕES 104

REFERÊNCIAS 106

APÊNDICE I – Pseudocódigo para geração de alternativas de manejo. 116

APÊNDICE II – Exemplo de funcionamento do algoritmo de geração de alternativas de manejo. 118

APÊNDICE III – Tabela de produção para os povoamentos *Pinus sp* em crescimento no final do horizonte de planejamento. 120

APÊNDICE IV – Pseudocódigo para o algoritmo *path* (basedo em McDill *et al.*, 2002). 121

APÊNDICE V – Modelo LINGO 123

APÊNDICE VI – Exemplo de agendamento com a melhor solução BT para o problema ARM_HP20 (D = desbaste, C = corte raso seguido de plantio). 124

APÊNDICE VII – Localização dos talhões nos quais o planejamento indica o corte raso nos períodos entre 1 e 7 anos, 8 e 14 anos e 15 e 20 anos, para o problema URM_HP20 com a melhor solução do algoritmo BT. 128

1. INTRODUÇÃO

Os modelos de planejamento florestal são ferramentas para identificação de planos que apresentam os melhores benefícios, geralmente econômicos, para o empreendimento florestal. Com estes é possível definir uma agenda de manejo para cada talhão que mantenha, adequadamente, a produção de madeira ao longo do tempo. Outra questão que também deve ser levada em conta é o impacto ambiental causado pelas ações de colheita. Geralmente estas preocupações são incorporadas ao processo de planejamento por meio de restrições espaciais ou de adjacência que, embora não abordem diretamente o problema, evitam características indesejáveis a um plano de colheita, como por exemplo, o corte raso de uma extensa área florestal.

Problemas de planejamento florestal com restrições de adjacência entre talhões são de difícil resolução e se constituem em um desafio para o manejador florestal (MURRAY, 1999). Em muitos casos, boas resoluções para os problemas podem levar horas, se não dias, para serem encontradas por meio das técnicas clássicas de pesquisa operacional (MCDILL E BRAZE, 2000). Este tipo de modelo inclui variáveis inteiras ou binárias, que se associam a problemas de natureza combinatória, de difícil solução por meio de técnicas exatas.

Os modelos de planejamento florestal com restrições de adjacência são classificados em dois tipos básicos, denominados por Murray (1999) de *unit restriction model* (URM) e *area restriction model* (ARM). No primeiro, a formulação do modelo se baseia na unidade de colheita, o que implica a geração de restrições considerando cada par de unidades adjacentes, enquanto no segundo, a restrição considera a área total de corte de unidades adjacentes.

Embora possíveis de serem resolvidos com o algoritmo exato *branch and bound* (RODRIGUES *et al*. 1998; SCHUCHOVSKI, 2003; SILVA *et al*., 2003 e SILVA *et al*., 2006), o tamanho potencial dos modelos de planejamento florestal com considerações espaciais pode tornar inviável o uso de tais procedimentos, sendo necessário, neste caso, recorrer ao uso de técnicas denominadas heurísticas ou meta-heurísticas, que, apesar de não garantirem a

solução ótima dos problemas, conseguem alcançar boas soluções, ou mesmo a solução ótima, em quantidade de tempo aceitável.

O termo meta-heurística foi utilizado pela primeira vez em 1986 no artigo que introduziu o algoritmo busca tabu (GLOVER, 1986). De acordo com Glover e Laguna (1997), meta-heurística se refere a uma estratégia mestra que guia e modifica outras heurísticas, para produzir soluções melhores que as normalmente geradas na busca do ótimo local.

O algoritmo de enxame de partículas (*Particle Swarm Optimization*, PSO) é uma técnica meta-heurística desenvolvida por Kennedy e Eberhart (1995). Ela se baseia no comportamento coletivo de animais, como por exemplo, bando de aves e cardumes de peixes. Neste algoritmo, cada solução do problema é representada por um elemento de uma população, denominada partícula, a qual se movimenta pelo espaço de busca do problema com base em sua própria experiência sobre as soluções encontradas e no conhecimento de toda a população de partículas.

A meta-heurística busca tabu (BT), criado por Glover (1986), ao contrário da PSO, baseia sua busca na vizinhança de apenas uma solução. Neste algoritmo, os movimentos realizados recentemente são armazenados em sua memória, na forma de uma lista tabu, e considerados proibidos por determinado tempo. Isto impede o retorno do algoritmo a locais previamente visitados, evitando a convergência para ótimos locais do espaço de busca.

As pesquisas com meta-heurísticas aplicadas ao planejamento florestal têm demonstrado o grande potencial destas técnicas. No entanto, a preocupação de garantir a sustentabilidade dos empreendimentos florestais tende a gerar modelos com horizontes de planejamento e quantidade de variáveis cada vez maiores. Isto implica em maior tempo computacional para encontrar soluções para os problemas.

Para contornar a demanda de tempo computacional em problemas de grande porte, a computação paralela aparece como uma ferramenta bastante promissora. A vantagem das meta-heurísticas PSO e BT é que estas podem ser facilmente paralelizáveis (MORAES, 2011 e FIECHTER, 1994). Nestes dois algoritmos, o cálculo da função objetivo é realizado de maneira independente podendo ser, dessa forma, eficientemente paralelizados.

Até o momento não foram encontradas implementações paralelas dos algoritmos PSO e BT na resolução de problemas de planejamento florestal. O algoritmo BT é uma técnica de alto desempenho, comprovadamente testado no planejamento florestal, enquanto a PSO ainda foi pouco explorada na resolução destes tipos de problemas. Neste sentido, esta pesquisa tem por objetivo apresentar soluções de implementação paralela e sequencial da PSO e da BT, em problemas de planejamento florestal com restrições espaciais.

2. OBJETIVOS GERAIS

Avaliar versões paralelas e sequenciais dos algoritmos de enxame de partículas e busca tabu aplicados a modelos de planejamento florestal com restrições de adjacência entre talhões a serem colhidos.

2.1. Objetivos específicos

- avaliar a formulação de modelos de planejamento florestal com as restrições de adjacência do tipo *unit restriction model* (URM) e *area restricition model* (ARM);
- comparar o desempenho das meta-heurísticas PSO e BT na resolução de problemas de planejamento florestal com restrições de adjacência entre talhões;
- avaliar duas diferentes formas de atualização de parâmetros da PSO sequencial;
- propor uma modificação ao algoritmo PSO para lidar com restrições de adjacência e compará-la ao algoritmo original e com a BT;
- comparar a versão paralela síncrona do algoritmo PSO com a versão sequencial do mesmo algoritmo na resolução de modelos de planejamento florestal;
- comparar a versão paralela do algoritmo BT com a versão sequencial do mesmo algoritmo na resolução de modelos de planejamento florestal;

3. REVISÃO DE LITERATURA

Neste item é apresentada uma revisão da literatura a respeito do planejamento florestal, das técnicas meta-heurísticas e sobre a computação paralela. Inicialmente, nos itens 3.1, 3.2 e 3.3, são abordados os temas a respeito do planejamento florestal, seus modelos e sobre os métodos de resolução destes problemas. No item 3.4 é discutido sobre meta-heurísticas, os algoritmos empregados e formas de avalição do desempenho destas. Por fim, o item 3.5 trata das questões referentes à computação paralela.

3.1. Planejamento florestal

Segundo Buongiorno e Gilles (1987), o gerenciamento florestal é a parte da ciência gerencial que estuda o aperfeiçoamento do processo de tomada de decisão, da ação e da avaliação das atividades econômicas, de curto e longo prazo, desenvolvidas no âmbito do setor de produção florestal. Um planejamento adequado é a base para o bom gerenciamento de recursos florestais. Megginson *et al.* (1998), definiram o planejamento como o processo de estabelecer objetivos ou metas, determinando a melhor maneira de atingi-las. O planejamento estabelece o alicerce para as subsequentes funções de organizar, liderar e controlar, e por isso é considerado função fundamental do gerenciador.

Megginson *et al.* (1998) apontaram algumas vantagens do planejamento, por exemplo: (1) ajuda a administração a adaptar-se e ajustar-se às condições mutáveis do ambiente; (2) ajuda a definir as responsabilidades com mais precisão; (3) dá mais ordem às operações; (4) tende a tornar os objetivos mais específicos e mais conhecidos; (5) diminui as conjecturas; (6) economiza tempo, esforço e dinheiro; e (7) ajuda a diminuir os erros na tomada de decisão.

De acordo com Johnston *et al.* (1977), no âmbito florestal o planejamento encontra mais problemas do que em muitos outros campos,

sobretudo, tratando-se de empresas de grandes dimensões, devido à longevidade própria das árvores, à extensão geográfica das áreas florestais e às incertezas da natureza e dos mercados.

Segundo Murray (1999), uma das maiores preocupações no desenvolvimento do planejamento da colheita dos talhões é identificar um plano no qual se produz os maiores benefícios, tipicamente, em termos econômicos. Ainda de acordo com este mesmo autor, outra preocupação é a dimensão temporal do planejamento, onde o fluxo contínuo de produção deve ser garantido, a fim de satisfazer as demandas do processo. A preocupação final é a de reduzir o impacto ambiental das atividades.

3.2. Pesquisa operacional

Segundo Ackoff e Sasieni (1971), a pesquisa operacional (PO) é a aplicação do método científico, por equipes interdisciplinares, a problemas que dizem respeito ao controle de sistemas organizados (homem-máquina), com a finalidade de obter as soluções que melhor satisfaçam aos objetivos da organização como um todo.

Dentre as inúmeras técnicas de PO existentes, sem dúvida a que mais se destaca é a programação linear (PL). De acordo com Goldbarg e Luna (2005), a PL constitui um caso particular dos modelos de programação em que as variáveis são contínuas e apresentam comportamento linear, tanto em relação às restrições quanto à função objetivo. A vantagem destes modelos está na extraordinária eficiência dos algoritmos de solução hoje existentes.

O principal algoritmo para solução de modelos de PL é o método Simplex. Segundo Goldbarg e Luna (2005), o algoritmo Simplex se destaca como uma das grandes contribuições à Programação Matemática deste século. De acordo com esses autores, trata-se de um algoritmo geral, extremamente eficiente para solução de sistemas lineares e adaptável ao cálculo computacional (apesar de algumas dificuldades clássicas).

Embora a importância da PL seja incontestável, o algoritmo Simplex é incapaz de lidar com variáveis inteiras, o que é justamente o tipo de variável

utilizada no planejamento florestal com considerações espaciais. Uma alternativa operacional para este inconveniente da PL seria o arredondamento da solução. De acordo com Lachtermacher (2007), uma das pressuposições da PL é a divisibilidade, ou seja, a pressuposição de que toda variável de decisão pode assumir qualquer valor fracionário. Lachtermacher (2007) afirma ainda que diversos problemas podem ocorrer pela utilização da técnica de arredondamento da solução, entre os quais: (1) nenhum ponto inteiro vizinho ao ponto ótimo é necessariamente viável e (2) mesmo que um dos pontos vizinhos seja viável, ele pode não ser necessariamente o ponto ótimo inteiro.

No planejamento florestal, são inúmeras as situações nas quais se exige a solução com variáveis inteiras. Rodrigues *et al.* (2003) apresentaram os seguintes exemplos: regulação florestal com escolha de um único regime por talhão, problemas de adjacência em talhões selecionados para colheita, seleção de rotas de transporte de madeira, seccionamento de toras, corte em indústria moveleira, corte de papel, dentre outros.

De acordo com Silva (2004), ao contrário da maioria dos problemas de PL, os problemas de programação linear inteira (PLI) têm um número finito de soluções. Silva (2004) afirma ainda que, paradoxalmente, é mais difícil encontrar a solução ótima de um problema de PLI do que de um de PL de forma que as soluções de modelos de PLI muitas vezes não podem ser encontradas por métodos exatos, exceto em problemas de pequeno porte. De acordo com Goldbarg e Luna (2005), o cerne da dificuldade dos problemas denominados *NP-hard*, que, por sinal, representam uma grande parte dos problemas de PLI realmente interessantes, está na explosão combinatória dos métodos enumerativos.

Um dos principais algoritmos de resolução de modelos de PLI é o método denominado *branch and bound* (B&B). Segundo Goldbarg e Luna (2005), este algoritmo consiste no desenvolvimento de uma enumeração inteligente dos pontos candidatos à solução ótima inteira de um problema. O termo *branch* refere-se ao fato de que o método efetua partições no espaço das soluções. O termo *bound* ressalta que a prova da otimalidade da solução utiliza-se de limites calculados ao longo da enumeração.

3.3. Modelos de planejamento florestal

Os modelos de planejamento florestal podem ser classificados basicamente em dois tipos, denominados por Johnson e Scheurman (1977) de modelo tipo I e modelo tipo II. De acordo com o Johnson e Scheurman (1997), os modelos I e II são caracterizados, basicamente, por quatro aspectos:

(1) O modelo tipo I preserva intacta ás áreas das unidades de manejo ao longo de todo o horizonte de planejamento (HP);

(2) No modelo tipo II, as áreas são colocadas em uma nova unidade de manejo cada vez que elas recebem corte raso;

(3) Ambos os modelos requerem uma restrição de área para cada unidade de manejo;

(4) Para cada unidade de manejo considerada no período um, uma atividade deve ser definida no modelo I para cada sequência possível de corte raso ao longo do HP. Por outro lado, no modelo II, para cada classe de idade no período um, uma atividade deve ser definida para cada duração de tempo até esta classe ser colhida ou deixada em estoque de produção ao final do HP. Para cada período, uma atividade também deve ser definida no modelo II para cada possível duração de tempo até a área ser novamente colhida ou deixada em estoque ao fim do HP.

Segundo Arce (2007), enquanto o modelo tipo I mantém a identidade dos talhões ao longo de todo o horizonte de planejamento, o modelo tipo II vai mesclando e recombinando as áreas cortadas em novos talhões, não necessariamente adjacentes, agrupados pela sua idade.

3.3.1. Considerações espaciais no planejamento florestal

Os modelos tipo I e II servem, em muitos casos, como base para novas formulações de modelos de planejamento florestal. Entre elas, as que incluem preocupações ambientais. Segundo Baskent e Keles (2005), os planos de

manejo com estas considerações incluem aspectos associados com a proteção da vida silvestre, biodiversidade, beleza cênica, redução da sedimentação pela água e erosão, entre outros. De acordo com estes autores, estes problemas consideram a forma e distribuição das unidades de manejo, restrições de adjacência, considerações sobre o tamanho máximo e mínimo de clareiras, conectividade, fragmentação, desenvolvimento de habitats de vários tamanhos, e considerações sobre estradas.

Segundo Barrett e Gilless (2000), as restrições de adjacência, em geral, definem limites máximos de unidades de colheita e especificam períodos de tempo, denominados de *green-up*, em que unidades adjacentes não podem ser colhidas. Diferentes períodos podem ser utilizados para o *green-up*. Van Deusen (1999) utilizou o período de dois anos. Em Liu *et al.* (2006) o valor de *green-up* utilizado foi de 15 anos, sendo que cada período do horizonte de planejamento tinha 20 anos. Lockwood e Moore (1992) o *green-up* foi de 20 anos considerando o planejamento com períodos de cinco anos.

De acordo com Bettinger e Boston (1999), no Estado de Oregon (EUA) a área máxima de corte raso é limitada em 48,56 ha (120 acres), além disto, talhões adjacentes não podem ser cortados até que as árvores atinjam a altura média de 1,35m ou 5 anos tenham passado desde a colheita do talhão adjacente. Ainda segundo estes autores, no Estado de Washington (EUA) as regras para o corte de talhões adjacentes é ainda mais complexa. A área máxima de corte raso é de 48,56 ha e os talhões devem atender uma das seguintes condições: (1) 90% do perímetro é composto de talhões com idade maior que 5 anos; (2) 60% do perímetro é composto de talhões com idade maior que 15 anos; e (3) 30% do perímetro é composto de talhões com idade maior que 30 anos. Segundo Dahlin e Sallnas[1] (1993), citado por Boston e Bettinger (1999), na região sub-alpina da Suécia as área de corte raso devem ser menores que 20 ha e unidades adjacentes não podem ser cortadas pelo período (*green-up* delay) de 15 anos.

[1] DAHLIN, B e SALLNAS, P. L. Harvest scheduling under adjacency constraints – A case study from Swedish sub-alpine region. **Scandinavian Journal of Forest Research**, v.8, p. 281-290, 1993.

Murray (1999) descreveu dois tipos básicos de restrições de adjacência, um baseado na unidade de manejo, denominado *unit restriction model* (URM) e outro baseado na área dos talhões, denominado *area restriction modelo* (ARM).

Para a utilização do modelo URM deve-se definir as unidades de manejo de tal forma que o corte de duas unidades adjacentes não viole a restrição espacial. Assim, esta abordagem requer que alguma técnica manual ou automática seja aplicada para delimitar as unidades de forma que se possa manter o limite de área máxima (MURRAY, 1999). De acordo com Richards e Gunn (2000), a vantagem da formulação URM é que estes modelos de restrição de adjacência são fáceis de formular e as soluções são facilmente visualizadas e entendidas. Por outro lado, estes autores afirmam que a grande desvantagem do modelo URM é que configurações alternativas de blocos de talhões são ignoradas no processo de otimização, levando possivelmente a soluções sub ótimas.

No modelo ARM as unidades de manejo são expressivamente menores que a área máxima e, assim, o corte de unidades adjacentes é permito sob a seguinte condição: a área total das unidades colhidas deve permanecer menor que o limite da restrição espacial (MURRAY, 1999).

Na formulação ARM apresentada em Lockwood e Moore (1993), a área total de cortes rasos é calculada para todos os talhões conectados ou adjacentes. Quando um bloco de talhões ultrapassa a área máxima de corte raso, utiliza-se uma função de penalização que transforma a violação em custo na função objetivo. Formulação semelhante foi empregada por Richards e Gunn (2000).

McDill *et al.* (2002) propuseram um algoritmo, denominado *path*, para a formulação do modelo ARM, este procedimento identifica todos os talhões adjacentes que juntos somam uma área limite, com as seguintes etapas:

(1) Inicie com qualquer par de talhões adjacentes. Se a área combinada destes talhões exceder a área máxima de colheita escreva a restrição de adjacência destes dois talhões. Este par de talhões forma o grupo (*cluster*) inicial de talhões processados, ou seja, o conjunto de talhões para os quais as restrições *path* já foram identificadas;

(2) Selecione qualquer talhão que é adjacente ao grupo atual de talhões processados e o adicione ao grupo;

(3) Defina uma rede, baseada no grupo atual, na qual os talhões são definidos como nós e o par de talhões adjacentes como arcos. Identifique cada caminho (*path*) possível originando do nó correspondente ao novo talhão do grupo. Termine o caminho quando a área acumulada dos talhões ao longo do caminho exceder a área máxima de colheita ou quando não há mais polígonos adjacentes ao caminho no grupo atual. Observe que o caminho pode seguir múltiplas ramificações, mas o caminho não pode retornar a um nó que já está no caminho. Quando um caminho excede a área máxima, determine se este é redundante. Um caminho recentemente criado é redundante se o conjunto de talhões em um caminho previamente identificado é subconjunto do grupo dos talhões do novo caminho. Se o novo caminho não é redundante adicione-o ao conjunto de restrições *path*.

(4) Pare quando o grupo (*cluster*) contém todos os talhões da floresta. Caso contrário, retorne ao passo (2).

Diferentes formulações para as restrições de adjacência foram propostas para o modelo URM. De acordo com McDill e Braze (2000), a motivação para o desenvolvimento de diferentes alternativas para estas formulações foi reduzir a quantidades de restrições nos modelos. Apesar disto, esses autores afirmaram que o número de restrições no problema não é necessariamente uma preocupação, pois existem atualmente inúmeros *softwares* que aceitam quantidade ilimitada de restrições nos modelos. Esses autores realizaram detalhada revisão sobre os tipos de restrições de adjacência URM, descrevendo-as em seis grupos básicos:

1) *Pairwise*: o tipo mais simples e mais amplamente aplicado. Como o nome já indica, restrições de adjacência *pairwise* requer uma restrição para cada par de talhões adjacentes em cada período.

2) *Modified Adjacency Matrix*: são baseadas na variação da matriz de adjacência. Uma matriz *M* x *M*, onde *M* é o número de talhões, cujos elementos a_{ij} são iguais a um (1) se os elementos *i* e *j* são adjacentes e igual a zero (0) caso contrário (os elementos na diagonal, a_{ii}, igual a zero).

3) *Type I*: são baseadas no conjunto de talhões mutuamente adjacentes, ou seja, conjuntos de talhões onde cada membro é adjacente a cada outro membro do grupo. Estas são consideradas restrições clique, ou seja, restrições onde a soma das variáveis binárias deve ser menor ou igual a um.

4) *Agregated Constraint group*: consiste de restrições que são formadas pela combinação do conjunto de restrições a fim de reduzir o número de restrições do modelo.

5) *Tightned Coefficient group*: são obtidas reduzindo alguns coeficientes de outro tipo de restrição de adjacência. Esta formulação resulta em uma solução ótima para o problema de programação linear relaxado, que é próxima a solução ótima da programação inteira-mista.

6) *Minimized Constraint group*: são obtidas pela resolução de um conjunto de problemas, que identifica o menor conjunto de restrições de adjacência entre todas as restrições.

Em relação às unidades de colheita dos modelos URM, nota-se que os autores Murray (1999) e McDill e Braze (2000) as descrevem de forma distinta. No primeiro, as unidades de colheita são formadas por grupos de talhões de tal forma que o corte de duas unidades adjacentes ultrapassa o limite de área. A formulação neste formato foi empregada por O'Hara *et al.* (1989). No entanto, os autores McDill e Braze descrevem os modelos URM de tal maneira que a unidade de colheita é o talhão e, assim, o corte de talhões adjacentes violam a restrição URM. A formulação descrita por McDill e Braze pode se tornar extremamente restritiva quando se tem uma floresta com talhões de tamanho reduzido.

3.3.2. Avaliação econômica de florestas

Segundo Rezende e Oliveira (2001), a avaliação econômica de projetos florestais envolve o uso de técnicas e critérios de análise que comparam os custos e as receitas inerentes ao projeto, visando decidir se este deve ou não ser implementado. No caso do planejamento florestal, a avaliação é feita para cada unidade de manejo em cada prescrição.

Um critério de avaliação bastante usual na avaliação econômica de florestas é o valor presente líquido (*VPL*). De acordo com Silva *et al.* (2005), o *VPL* é a diferença do valor presente das receitas menos o valor presente dos custos. Esses autores afirmaram que este método traz implicitamente a consideração do tamanho do projeto ou o volume de capital investido. Porém, o método não considera o horizonte do projeto, havendo a necessidade de corrigir os horizontes com diferentes durações. Segundo Clutter *et al.* (1983), se o *VPL* é positivo, o investimento gerará lucro após o pagamento do capital e do custo dos juros incorridos. Se for negativo, o retorno do investimento não será suficiente para pagar o capital investido e os juros da taxa utilizada.

Nos anos finais do horizonte de planejamento nos modelos de planejamento florestal é comum que os povoamentos tenham florestas em crescimento. Estas florestas não podem ser avaliadas da mesma forma daquelas em idade de rotação. Schneider e Durlo (1987) descreveram três formas de avaliar estas florestas:

1) *Valor da exploração*: refere-se ao valor comercial do estoque de madeira ou parte da mesma menos os custos de colheita. Este método de determinação pode ser feito para qualquer povoamento que alcançou ou está perto da idade de corte final.

2) *Valor do custo do povoamento*: para os povoamentos jovens não serve a determinação do valor da exploração, pois os custos são mais elevados que a renda. Igualmente a avaliação segundo o valor da expectativa da produção também apresenta falhas, pois há incerteza quanto ao desenvolvimento do povoamento até então imprevisível. Em função disto, a melhor forma de avaliar estes povoamentos é através de seus custos de implantação.

3) *Valor da expectativa de produção*: A idade correta para avaliação de um povoamento por este método é quando este se encontra próximo à idade de rotação. O valor da expectativa da produção de um povoamento é composto por todas as receitas menos os custos, que se podem esperar desde o momento da avaliação até o final da rotação, inclusive a renda do corte final, tudo estimado para o ano do fim da rotação e descapitalizado para o momento de avaliação.

3.4. Meta-heurísticas

Uma heurística é uma técnica que busca alcançar uma boa solução utilizando um esforço computacional considerado razoável, sendo capaz de garantir a viabilidade ou a otimalidade da solução encontrada ou, ainda, em muitos casos, ambas, especialmente nas ocasiões em que essa busca partir de uma solução viável próxima ao ótimo (GOLDBARG e LUNA, 2005).

De acordo com Voss (2001), heurística é uma técnica (que consiste em uma regra ou um conjunto de regras) que busca (com a expectativa de encontrar) boas soluções com custo computacional razoável. Ainda segundo Voss (2001), a heurística é aproximativa no sentido que fornece uma boa solução com pequeno esforço relativo, mas não garante otimalidade.

As meta-heurísticas referem-se a uma estratégia que guia e modifica outras heurísticas, fornecendo soluções melhores que aquelas normalmente geradas em uma busca de otimalidade local (GLOVER e LAGUNA, 1997).

Como forma de sumarizar as diversas questões que envolvem as meta-heurísticas, Voss *et al.* (1999) citado por Voss (2001) sugerem a seguinte definição: Meta-heurística é um processo mestre iterativo que guia e modifica as operações de heurísticas subordinadas, para eficientemente produzir soluções de alta qualidade. A heurística subordinada pode ser um procedimento de alto ou baixo nível, ou uma simples busca local, ou um método de construção. A família de meta-heurísticas inclui, mas não é limitada a, procedimentos de memória adaptativa, busca tabu, sistemas de formigas, *greedy randomized adaptive search*, busca em vizinhança variável (*variable neighborhood search*), métodos evolucionários, algoritmos genéticos, *scatter search*, redes neurais, *simulated annealing* e seus híbridos

3.4.1. Otimização por enxame de partículas (PSO)

O algoritmo enxame de partículas (*particle swarm optimization* - PSO) foi criado por Kennedy e Eberhart (1995), inspirados pelas teorias da inteligência

coletiva, a qual trata de desenvolvimento de sistemas baseados no comportamento coletivo de insetos, tais como as formigas, cupins, abelhas, entre outros, assim como de outros animais que vivem em grupo tais como os bandos de pássaros e cardumes de peixes.

Em relação aos cardumes de peixes, Wilson (1975)[2], citado por Kennedy e Eberhart (1995), afirmou que na natureza, em teoria ao menos, membros individuais de um cardume podem se beneficiar da descoberta e da experiência prévia de todos os outros membros do cardume durante a procura de alimento. De acordo com Kennedy e Eberhart (1995), esta afirmação sugere que o compartilhamento social da informação entre os indivíduos oferece uma vantagem evolucionária.

Outros algoritmos também foram determinantes para o desenvolvimento do algoritmo PSO. O primeiro deles foi o *Boids* desenvolvido por Reynolds (1987). Este simula em computador o comportamento em grupo de animais, tais como um cardume e um bando de pássaros e consiste de três regras básicas: (1) os pássaros (ou *Boids* como o autor os denominou) devem evitar colidir com seus vizinhos; (2) devem buscar igualar com a velocidade e direção de voo dos outros pássaros; e (3) eles devem procurar permanecerem pertos uns dos outros. O desenvolvimento do algoritmo *Boids* tomou como base as ideias do algoritmo que simula um sistema de partículas desenvolvido por Reeves (1983). O sistema de Reeves (1983) descreve um método para simular em computador objetos indistintos, tais como nuvens e explosões. Segundo Banks *et al.* (2007), sistemas como este são amplamente utilizados atualmente na geração de efeitos especiais do cinema e ambientes gráficos interativos realistas.

O algoritmo PSO foi concebido então levando em conta os conceitos de inteligência coletiva, do algoritmo *Boids* e, por fim do algoritmo de Heppner e Grenader (1990), o qual fornece um objetivo de busca por comida para os pássaros. Este último fornecendo a ideia básica para ligar o movimento das partículas às funções objetivo das formulações matemáticas dos problemas.

Segundo Castro e Tsuzuki (2007), a PSO é uma técnica que se baseia no movimento coletivo de um grupo de partículas: o enxame de partículas.

[2] WILSON, E. O. **Sociobiology: The new synthesis**. Cambridge, MA: Belknap Press, 1975.

Cada membro deste enxame é movimentado através do espaço de busca do problema por duas forças. Uma os atrai com uma magnitude aleatória para a melhor localização já encontrada por ele próprio, e outra para a melhor localização encontrada entre alguns ou todos os membros do enxame. A posição e a velocidade de cada partícula são atualizadas a cada iteração até todo o enxame convergir.

Cada partícula possui dois componentes básicos: a posição em que se encontra no espaço de busca (x_p) e a velocidade (v_p). No algoritmo original (KENNEDY e EBERHART, 1995) a cada iteração estes dois componentes são atualizados pelas expressões (1) e (2).

$$x_p^{(it+1)} = x_p^{(it)} + v_p^{(it+1)} \tag{1}$$

$$v_p^{(it+1)} = v_p^{it} + c_1 \cdot rand_1^{(it)} \cdot \left(pbest_p^{(it)} - x_p^{(it)}\right) + c_2 \cdot rand_2^{(it)} \cdot \left(gbest_p^{(it)} - x_p^{(it)}\right) \tag{2}$$

em que: p: partícula; v_p : velocidade da partícula p; x_p : posição da partícula p; c_1: coeficiente cognitivo; c_2: coeficiente social; $rand$: função aleatória de distribuição uniforme entre 0 e 1; $pbest_p$: melhor posição que a partícula p já obteve durante a busca; $gbest_p$: melhor posição encontrada na vizinhança da partícula p; e it: iteração atual.

A posição x_p de cada partícula é composta de um conjunto de coordenadas que representa uma solução pontual dentro do espaço de busca. A cada iteração esta posição é avaliada de acordo com uma função de aptidão (*fitness*). Se esta posição for melhor que a já encontrada, ela é armazenada em $pbest_p$.

Cada partícula pertence a uma vizinhança social, desta forma seu comportamento é afetado por boas posições encontradas por suas vizinhas. A melhor posição encontrada na vizinhança de uma partícula é representada por $gbest_p$. Desta forma, a movimentação de cada partícula durante a execução do algoritmo dependerá da sua velocidade anterior, e das posições $pbest_p$ e $gbest_p$ (MENDES, 2004).

Na FIGURA 1 é apresentado um exemplo do comportamento das partículas durante uma iteração do algoritmo PSO, conforme a implementação original de Kennedy e Eberhart (1995).

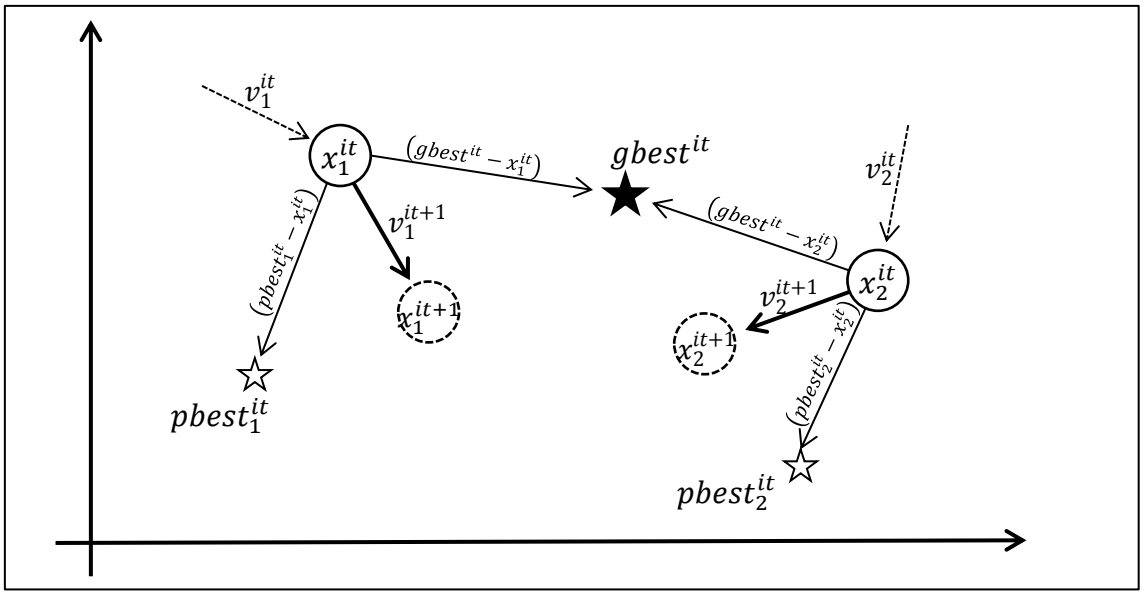


FIGURA 1 - COMPORTAMENTO DAS PARTÍCULAS EM ESPAÇO BIDIMENSIONAL (ADAPTADO DE MÜLLER, 2007).

3.4.1.1. Parâmetros da PSO

De acordo com Kennedy *et al.* (2001), no enxame de partículas existem vários parâmetros explícitos cujos valores podem ser ajustado para produzir variações na forma como o algoritmo realiza a busca no espaço do problema. Os mais importantes são V_{max} e as constantes de aceleração (c_1 e c_2), que são fixados no início do algoritmo e são mantidos constantes durante a execução. Ainda segundo esses autores, as manipulações destes parâmetros podem causar grandes mudanças no comportamento do sistema.

Segundo Sierakowski (2006), os coeficientes c_1 e c_2 são constantes positivas denominadas de componente de cognição e componente social, respectivamente. Estas são as constantes de aceleração que variam a velocidade de uma partícula em direção ao *pbest* e *gbest*, de acordo com a experiência passada. Ainda segundo Sierakowski (2007), não são fatores

críticos para determinar a convergência do algoritmo. Porém, um ajuste correto destes valores pode levar a uma convergência mais rápida do algoritmo.

A configuração c_1 = c_2 = 2, sugerida por Kennedy *et al.* (2001), parece ser a mais utilizada na literatura (YOSHIDA *et al.*, 2001; NAKA, 2001; SANDRINI, 2005; TEBALDI *et al.*, 2006; SIERAKOWSKI, 2006; NASCIMENTO *et al.*; 2007; NASCIMENTO *et al.*, 2008; SIERAKOWSKI e COELHO, 2008; NASCIMENTO, *et al.*, 2012 e NASCIMENTO *et al.*, 2013). Valores diferentes foram usados por Medeiros (2005), c_1 = c_2 = 1,5, e Sha e Hsu (2008), c_1 = 0,7 e c_2 = 0,1. Nos trabalhos de Shan *et al.* (2012) e Smythe (2012) diversas configurações para estes parâmetros foram testadas.

De acordo com Kennedy *et al.* (2001), o algoritmo PSO tem a tendência de explodir em oscilações cada vez mais amplas a medida que se aumentam as iterações, a menos que algum método seja aplicado para restringir a velocidade. De acordo com esses autores o método mais usual para prevenir a explosão é simplesmente definir um parâmetro de velocidade máxima (V_{max}) e impedir que a velocidade exceda este parâmetro em cada dimensão para cada partícula:

$$se\ v_p > V_{max}\ entao\ v_p = V_{max}\ senão\ se\ v_p < -V_{max}\ então\ v_p = -V_{max}$$

em que V_{max}: velocidade máxima de uma partícula em cada iteração.

Segundo Mendes (2004), se V_{max} for configurado com um valor alto as partículas podem passar despercebidas por boas soluções. Do contrário, se V_{max} tiver um valor baixo as partículas explorarão muito lentamente, e boas soluções podem não ser encontradas. Neste caso, as partículas podem vir a ficar presas em ótimos locais.

Na literatura pesquisada o valor para V_{max}, quando expecificado, foi sempre configurado entre 10 e 20% (SANDRINI, 2005; SIERAKOWSKI, 2006; TEBALDI *et al.*, 2006; NASCIMENTO *et al.*, 2008; SIERAKOWSKI e COELHO, 2008; NASCIMENTO, *et al.*, 2012 e NASCIMENTO *et al.*, 2013).

Quanto ao tamanho da população, Kennedy *et al.* (2001) afirmaram, com base em suas experiências, que tamanhos entre 10 e 50 normalmente parecem trabalhar melhor. Segundo Mendes (2004), a PSO necessita de

populações menores que outros algoritmos evolucionários para alcançar soluções de qualidade elevada.

Populações pequenas, menores que 100 partículas, parecem ser mais usuais (YOSHIDA *et al.*, 2001; NAKA, 2001; SANDRINI, 2005; TEBALDI *et al.*, 2006; SIERAKOWSKI, 2006; NASCIMENTO *et al.* 2007; SHA e HSU, 2006; SIERAKOWSKI e COELHO, 2008; NASCIMENTO, *et al.*, 2012 e NASCIMENTO *et al.*, 2013). No entanto, outros autores conseguiram melhores resultados testando populações maiores, como por exemplo, os trabalhos de Medeiros (2005), Shan *et al.* (2012) e Smythe (2012).

3.4.1.2. Modificações ao algoritmo PSO

Desde a sua criação em 1995, diversos pesquisadores têm sugerido modificações ao algoritmo PSO original. Dentre estas, cabe destacar duas abordagens que atualmente podem ser consideradas como versões clássicas do algoritmo PSO, sendo que as demais proposições de modificações posteriores ao algoritmo se baseiam em um destes componentes: momento de inércia (SHI e EBERHART, 1998) e fator de constrição (CLERC e KENNEDY, 2002).

O parâmetro momento de inércia (w) tem o objetivo de melhorar o controle sobre o processo de busca do algoritmo e reduzir, ou até eliminar, a importância do V_{max}. De acordo com o que foi proposto por Eberhart e Shi (1998) a equação de velocidade original, expressão (1), é alterada conforme a expressão (3).

$$v_p^{(it+1)} = w \cdot v_p^{it} + c_1 \cdot rand_1^{(it)} \cdot \left(pbest_p^{(it)} - x_p^{(it)}\right) + c_2 \cdot rand_2^{(it)} \cdot \left(gbest_p^{(it)} - x_p^{(it)}\right) \quad (3)$$

em que: w: momento de inércia.

De acordo com Eberhart e Shi (2000), melhores desempenhos têm sido encontrados configurando inicialmente w com um valor relativo alto (0,9, por exemplo), o que corresponde a maiores áreas de busca, e gradualmente

diminuindo w para menores valores (0,4, por exemplo), onde a busca passa a ser direcionada a encontrar um ótimo local.

O coeficiente denominado fator de constrição foi desenvolvido por Clerc e Kennedy (2002), como forma de limitar a movimentação das partículas. Desta forma, a equação de velocidade original, expressão (1), é modificada conforme a expressão (4).

$$v_p^{(it+1)} = \chi \cdot \left[v_p^{it} + c_1 \cdot rand_1^{(it)} \cdot \left(pbest_p^{(it)} - x_p^{(it)} \right) + c_2 \cdot rand_2^{(it)} \cdot \left(gbest_p^{(it)} - x_p^{(it)} \right) \right] \quad (4)$$

em que: χ: coeficiente de constrição.

O coeficiente de constrição é calculado conforme a expressão (5).

$$\chi = \frac{2 \cdot k}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi} \right|} \quad (5)$$

em que: $\varphi = c_1 + c_2$, $\varphi > 4$.

Segundo Mendes (2004) a maioria das pesquisas usando o método de constrição costuma configurar φ igual a 4,1 (tendo assim $c_1 = c_2 = 2{,}05$) e $k = 1$ o que determina um $\chi \approx 0{,}729$. Isto é algebricamente equivalente a usar o momento de inércia com $w = 0{,}729$ e $c_1 = c_2 \approx 1{,}49445$.

3.4.1.3. Aplicações do algoritmo PSO no planejamento florestal

O algoritmo PSO começou a ser aplicado recentemente a problemas de planejamento florestal, com o primeiro trabalho de Brooks (2010). Desde então algumas pesquisa têm sido conduzidas no sentido de superar as dificuldades do algoritmo em lidar com as relações de adjacência entre os talhões. Três tipos de modificações para lidar com este problemas foram propostas: (1) abandonar as partículas com soluções infactíveis (SHAN *et al.*, 2012); (2) usar uma função que iterativamente considera cada talhão em uma solução, e sempre que uma violação de adjacência é encontrada, compara-se o volume de colheita do talhão atual em relação ao volume total dos seus vizinhos. Se o

volume do talhão atual for maior, então os adjacentes são definidos para não-corte (SMYTHE *et al.*, 2012); e (3) funções de penalidade (SHAN *et al.* 2012 e SMYTHE *et al.*, 2012), método clássico frequentemente utilizado com as meta-heurísticas.

A fim de melhorar o desempenho da PSO em problemas de planejamento florestal com restrições espaciais, alguns autores propuseram modificações profundas ao algoritmo original, obtendo resultados de qualidade semelhante a outras meta-heurísticas. Brooks (2010) propôs uma implementação da PSO denominada *priority representation* PSO (PPSO) na qual cada posição no vetor de posições representa um item discreto a ser incluída em uma permutação. Esta permutações são feitas de tal forma que garantem que algumas restrições, ou mesmo todas, não sejam violadas. Smythe *et al.* (2012) propõem e implementam uma variante do algoritmo PSO denominada *Roulette Wheel* PSO (RWPSO). Neste algoritmo proposto, a localização de uma partícula em cada dimensão é gerada por uma probabilidade de roleta das velocidades em cada dimensão do problema.

No trabalho de Shan *et al.* (2012) o algoritmo PSO foi comparado com a solução obtida por métodos exatos, obtendo 86% da solução ótima global, quando o algoritmo inicia em posições com soluções factíveis. Estes autores afirmaram que algoritmo PSO quando inicializado com a posição das partículas de forma aleatória pode ter aplicação limitada em problemas de planejamento florestal com objetivos econômicos, restrições de fluxo de produção e considerações espaciais. Por outro lado, em Nascimento *et al.* (2012) o algoritmo PSO foi avaliado em um problema de planejamento florestal sem as considerações espaciais e apresentou 99,07% do ótimo matemático.

3.4.2. Busca tabu

De acordo com Glover (1989), o algoritmo busca tabu (BT) é um procedimento adaptativo com a capacidade de fazer uso de diversos outros métodos, tais como programação linear e heurísticas especializadas, os quais são direcionados para superar a otimalidade local. Glover (1989) afirma

também que o algoritmo possui dois elementos básicos: um que restringe a busca de soluções classificando certos movimentos como proibidos, ou tabus; e outro que libera a busca providenciando “esquecimentos estratégicos”, denominado de critério de aspiração.

De acordo com Bettinger *et al.* (1997), a BT foi desenvolvida a partir de técnicas de busca em gradiente, as quais garantem a solução ideal quando o espaço de busca é convexo. Problemas de planejamento florestal frequentemente não possuem espaço de busca convexo e muitos têm variáveis discretas. Ao contrário de heurísticas mais simples, tais como o *hill climbing*, a BT possui a habilidade de continuar a busca de novas soluções sem ser confundida pela ausência de movimentos que melhoram a solução e sem voltar para um ótimo local previamente visitado (GLOVER, 1989).

Segundo Richards e Gunn (2000), a essência da filosofia da BT é misturar elementos das técnicas do cérebro humano para resolução de problemas na estrutura da otimização matemática. O conceito fundamental usado em toda BT é a implementação de estruturas de memória, as quais armazenam informações sobre o histórico de busca. A mais básica destas é a lista tabu, a qual registra a condição tabu dos movimentos e é denominada de memoria de curto prazo. Outras estruturas de memoria, as quais refletem o médio e o longo prazo do histórico de busca são usados para o desenvolvimento de algoritmos que propõem diversificação e intensificação da busca para melhorar a exploração do espaço de busca.

3.4.2.1. Movimentos e vizinhança na busca tabu

Dois aspectos críticos ao BT e que podem ser determinantes para o desempenho do algoritmo são a forma como são realizados os movimentos e como é definida a vizinhança de uma dada solução. Segundo Bettinger *et al.* (1999), movimentos consistem em modificações potenciais a uma solução corrente e são caracterizados pelo seu impacto no valor da função objetivo, quando uma alternativa de manejo de um talhão é modificada. Uma vizinhança

consiste em todas as possíveis mudanças em uma solução, por meio de movimentos potenciais que modificam uma solução corrente.

Conforme Rodrigues *et al.* (2003), os movimentos são utilizados para fugir de ótimos locais e devem ser hábeis para aceitar uma solução mesmo que esta tenha qualidade inferior à solução atual. Estes autores descrevem dois tipos de vizinhança: (1) na vizinhança total o mecanismo de movimento efetua todas as trocas de pares de variáveis (X_{ij}), transformando uma variável não-básica ($X_{ij} = 0$) em variável básica ($X_{ij} = 1$); e (2) na vizinhança parcial apenas uma porção da vizinhança total, escolhida aleatoriamente, é avaliada a cada iteração. Para o problema de planejamento florestal, onde somente uma alternativa de manejo pode ser escolhida para cada talhão, a adição de uma variável na base ($X_{ij} = 1$) implica a remoção de outra variável da base ($X_{ij} = 0$). É importante ressaltar que usar todas as mudanças possíveis para gerar a vizinhança só é aceitável quando há um número reduzido de alternativas de manejo para cada talhão.

Em pesquisas do algoritmo BT no planejamento florestal é comum a utilização de métodos para intensificação da busca. Uma abordagem frequente é utilizar os movimentos denominados 1-opt e 2-opt. No primeiro, a solução vizinha tem apenas um talhão onde se modificada a alternativa de manejo. No segundo, a troca é feita em dois talhões, podendo ser feita independentemente ou trocando a alternativa de um talhão pela aquela de outro talhão. Esta abordagem foi empregada nos algoritmos BT de Bettinger *et al.* (1999), Boston e Bettinger (2002) e Pukkala e Heinonen (2004).

Outra questão importante na movimentação do algoritmo é lidar com soluções infactíveis. Uma forma de evitar as restrições de adjacência é considerar a alternativa de manejo de não realizar nenhuma colheita do talhão ao longo do horizonte de planejamento, tal como foi utilizado por Gomide *et al.* (2013). No algoritmo BT desenvolvido por Bettinger *et al.* (1997), a movimentação é realizada de uma solução factível para outra solução factível. Bettinger *et al.* (1997) usaram a lógica na qual, um movimento pode consistir das seguintes operações: selecionar um talhão para colheita (em qualquer período), desmarcar um talhão para colheita (todos os períodos sem colheita) e trocar o período no qual um talhão é colhido. Um movimento candidato não

pode marcar a colheita de um talhão em dois ou mais períodos e colher talhões adjacentes no mesmo período.

3.4.2.2. Lista tabu e tempo tabu

Um elemento comum a todos os algoritmos busca tabu é a lista tabu. De acordo com Bettinger *et al.* (1997), a lista tabu é chamada de memória de curto prazo (*short-term memory*). Segundo Richards e Gunn (2000), movimentos anteriormente visitados são proibidos, ou tabus, por *T* iterações. *T* é denominado tempo tabu (*tabu tenure*). Segundo estes autores, quando movimentos realizados recentemente são considerados tabu, a busca é forçada a encontrar diferentes soluções para o problema, a qual irá ter uma nova vizinhança e consequentemente um novo ótimo local pode ser encontrado.

Segundo Pukkala e Heinonen (2004), o algoritmo busca tabu memoriza movimentos recentes proibindo que sejam usados por algum tempo. Tipicamente, um movimento é mantido na lista tabu certo número de iterações. Este número é o tempo tabu inicial do movimento. Cada iteração reduz o tempo tabu de todos os movimentos. Um movimento é novamente permitido quando seu tempo tabu diminuiu para zero.

3.4.2.3. Critério de aspiração

O critério de aspiração é quando se aceita um movimento considerado tabu por apresentar valor da função objetivo melhor que o já encontrado até o momento (BETTINGER *et al.*, 1999). Glover (1989) denominou este aspecto do algoritmo de “esquecimentos estratégicos”.

Há também os casos em que todas as soluções candidatas estão na lista tabu. Pukkala e Heinonen (2004) empregaram o critério de escolher

aquela com o menor tempo tabu. Já o mecanismo implementado por Bettinger e Boston (1999) escolhe a solução de melhor qualidade.

3.4.2.4. Memória de longo prazo

Segundo Bettinger *et al.* (1997), na memória de longo prazo (*long-term memory*), o algoritmo armazena o número de vezes que uma solução vizinha entrou ou saiu da solução atual, quando estas atingem um determinado número são penalizadas de forma a impedir que sejam aceitas

Bettinger e Boston (1999) avaliaram uma versão do algoritmo BT empregando uma memória tabu de longo prazo. Nesta, armazena-se a frequência com que cada movimento é inserido na lista tabu. Depois de 3.000 iterações o algoritmo gera uma nova solução utilizando os movimentos menos selecionados.

3.4.2.5. Modificações e melhorias ao algoritmo BT

Uma modificação do algoritmo busca tabu aplicada ao planejamento florestal foi utilizada por Richards e Gunn (2000), na qual o tempo tabu é automaticamente atualizado ao longo da execução do algoritmo. Esta modificação é denominada busca tabu reativa (*reactive tabu search* - RTS) e foi proposta por Batitti e Tecchiolli (1994)[3]. No inicio o tempo tabu *T* é configurado em uma iteração. O valor de T é aumentado quando soluções repetem na trajetória da busca e diminui quando repetições não ocorrem por uma determinada quantidade de movimentos. Assim, o sistema explicitamente detecta ciclos por soluções previamente encontradas e aumenta o tempo tabu melhorando a diversificação da busca.

[3] BATITTI, R. e TECCHIOLLI, G. The reactive tabu search. **ORSA J. Comput**, v. 6, n. 2, p. 126-140, 1994.

3.4.2.6. Aplicações da busca tabu no planejamento florestal

Nas pesquisas em que a BT foi aplicado ao planejamento florestal e comparado com a solução ótima global, o algoritmo sempre conseguiu alcançar soluções próximas ao da solução pelo método exato. Entre 93,7% e 100% em Bettinger e Boston (1999), entre 94,74% e 99,13% em Bettinger *et al.* (1999) e entre 90,39% e 98,84% em Rodrigues *et al.* (2003). Quando comparada com outras meta-heurísticas, a BT aparece frequentemente como a melhor opção, ou entre as melhores técnicas avaliadas (BETTINGER e BOSTON, 1999; BETTINGER *et al.*, 2002; PUKKALA e HEINONEN, 2004).

Quanto às abordagens 1-opt e 2-opt, as pesquisa apontam que versão 2-opt tende a encontrar melhores resultados (BETTINGER *et al.*, 1999; BOSTON e BETTINGER, 2002; e PUKKALA e HEINONEN, 2004)

3.4.3. Avaliação de meta-heurísticas

O objetivo da avalição das meta-heurísticas é responder as perguntas: qual é o melhor algoritmo, ou qual é a melhor configuração de um algoritmo, para resolução de um determinado problema. De acordo com Eberhart e Shi (2007), a resposta é simples quando todos os valores de *fitness* de uma configuração são melhores do que os de outra. No entanto, raramente a solução é tão simples. Principalmente nas fases posteriores ao desenvolvimento do algoritmo, quando se faz o ajuste fino de parâmetros a fim de maximizar seu desempenho.

Mesmo sendo possível determinar o quão bom (ou mal) é o desempenho de diferentes algoritmos, é necessário afirmar se a diferença entre os métodos é estatisticamente significativa. Segundo Eberhart e Shi (2007), deve-se ter cautela ao escolher a ferramenta estatística a ser utilizada, uma vez que, os conjuntos de dados de meta-heurísticas, tais como listas de *fitness*, normalmente não se ajustam a qualquer tipo particular de distribuição. Neste caso o recurso é fazer uso de estatísticas não paramétricas.

Segundo Zar (2010), existe uma grande quantidade de métodos estatísticos que não exigem a estimativa de parâmetros populacionais (como a média e o desvio padrão). Estes procedimentos estatísticos são denominados testes não paramétricos.

Em um levantamento feito por Derrac *et al.* (2011), verificou-se diversos procedimentos não paramétricos que podem ser empregados para comparação de meta-heurísticas. Segundo estes autores, como alternativa ao teste ANOVA, os testes de Friedman, Iman-Davenport, Friedman Aligned e Quade podem ser utilizados. No caso dos testes *post-hoc*, este autores sugerem os procedimentos Benferroni-Dunn, Holm, Hochberg, Hommel, Holland, Rom, Finner e Li para comparações de um algoritmo de controle em relação aos demais algoritmos, e os procedimentos Nemenyi, Holm, Shaffer e Bergman para comparações múltiplas entre todos os algoritmos avaliados.

Dentre os métodos mais empregados como alternativa a ANOVA para avaliação de meta-heurísticas pode-se citar o teste de Friedman (NÁPOLES *et al.*, 2012; SABAT *et al.*, 2011 e GARCÍA et al, 2009) e Kruskall-Wallis (GOMIDE *et al.*, 2009a e NASCIMENTO *et al.*, 2012). Quanto aos testes *post-hoc* podem ser mencionados o teste de U de Mann-Whitney (EBERHART e SHI, 2007), teste de Wilcoxon (NÁPOLES *et al.*, 2012) e Teste de Dunn ou Benferroni-Dunn (SABAT *et al.*, 2011; GARCÍA *et al.*, 2009 e NASCIMENTO *et al.*, 2012). Nestas pesquisas o nível de significância utilizado foi sempre de 0,05.

Em relação aos trabalhos voltados a resolução de problemas de planejamento florestal, Bettinger *et al.* (2009) fizeram levantamento dos procedimentos utilizados para avaliação e validação de meta-heurísticas. Estes autores sugerem seis níveis de validação que vão desde a não avaliação do desempenho, até a comparação com a solução ótima alcançada por meio de técnicas exatas. Estes níveis são descritos a seguir.

(1) Nenhuma validação ou desempenho é estabelecido: em casos em que o modelo de planejamento florestal envolve dificuldades devido ao tamanho e complexidade e, desta forma, a verdadeira solução ótima é desconhecida;

(2) A auto avaliação é estabelecida: estatísticas básicas tais como média e desvio padrão são aplicadas para avaliar a qualidade das soluções geradas

pela meta-heurística. Neste caso um conjunto de amostras de soluções heurísticas é requerido, cada solução com um diferente valor da função objetivo;

(3) Comparação com a solução de outras meta-heurísticas: a comparação pode envolver uma meta-heurística "*standard*" com qualidade comprovada anteriormente ou ainda um grupo de meta-heurísticas em que se deseja escolher a mais adequada a determinado problema;

(4) Comparação com a solução global ótima estimada: utiliza-se a teoria dos valores extremos para gerar uma solução ótima global estimada. Por meio do ajuste estatisticamente significativo com os dados da amostra para uma curva *Weibull* pode-se utilizar o parâmetro de localização para estimar a solução global para o problema de planejamento florestal em questão;

(5) Comparação com a solução ótima gerada por problemas similares: aplicada em situações onde há problemas similares que podem ser resolvidos com métodos que gerem a solução ótima. Nestes casos, as restrições dos problemas são relaxadas (uma ou mais restrições são ignoradas) ou o problema é otimizado baseado em modelos determinísticos de simulação;

(6) Comparação com a solução gerada por técnicas exatas tais como a programação inteira ou inteira mista, ou ainda, pela completa enumeração das soluções: é a abordagem mais utilizada para a validação de meta-heurísticas. Neste caso, a solução ótima é gerada, geralmente, ao custo de elevados tempos computacionais, e a solução da meta-heurística é então comparada com ela.

3.5. Computação paralela

A computação paralela teve início na década de 1980, quando diversos fabricantes começaram a produzir computadores que exploravam algum tipo de arquitetura de paralelismo. Porém, nas décadas seguintes, o avanço no desempenho dos processadores focou no aumento da velocidade do *clock* (pulso elétrico que sincroniza as atividades do computador). Esta abordagem, no entanto, tinha suas limitações, particularmente no que diz respeito ao

consumo de energia e emissão de calor. Diante disso, nos últimos anos, os fabricantes voltaram para as ideias da década de 1980: vários processadores que compartilham memória são configurados em uma única máquina e, cada vez mais, em um chip, abordagem conhecida como *multicore* (CHAPMAN *et al.*, 2008).

Embora a tecnologia da computação paralela já exista a várias décadas, ela só tornou-se disponível para notebooks e desktop a partir de 2005 com o lançamento da geração de processadores Athlon™ 64 X2 o primeiro processador x86 *dual-core* (AMD, 2014) e posteriormente em 2006, com o lançamento da geração de processadores Core™ 2 Duo (INTEL, 2014).

De acordo com Moraes (2011), a grande limitação para a utilização dos métodos não determinísticos, em particular de meta-heurísticas, como o algoritmo genético e o enxame de partículas é o elevado número de avaliações da função objetivo e restrições. Quando o tempo computacional de avaliação destas funções é elevado, a aplicação destes métodos pode conduzir a tempos de processamento extremamente elevados e no caso das funções que envolvem a simulação de problemas reais de engenharia a sua aplicação pode ser inviável. Neste sentido, a computação paralela ou processamento paralelo parece ser uma das alternativas mais viáveis atualmente para lidar com problemas de otimização de grande porte.

3.5.1. API para desenvolvimento de aplicativos paralelos

Com o avanço dos computadores paralelos tornou-se necessário o desenvolvimento de API (*Application Programming Interface*) específicas para o desenvolvimento de programas paralelos. Segundo Chapman *et al.* (2008), as API consistem em um conjunto bem definido de recursos de linguagem, rotinas de biblioteca, anotações, ou diretrizes que podem ser empregadas por um programador, muitas vezes servem para esconder detalhes de implementação de nível mais baixo do usuário. Dentre as API mais utilizadas para o desenvolvimento de código paralelo tem-se a OpenMP (*Open Multi-Processing*), baseada em memória compartilhada e aquelas implementadas

com base no padrão MPI (*Message Passing Interface*) como por exemplo a Open MPI e a MPICH, as quais são voltadas para as arquiteturas de memória distribuída. Na FIGURA 2 são apresentadas representações de arquiteturas de memória compartilhada e distribuída.

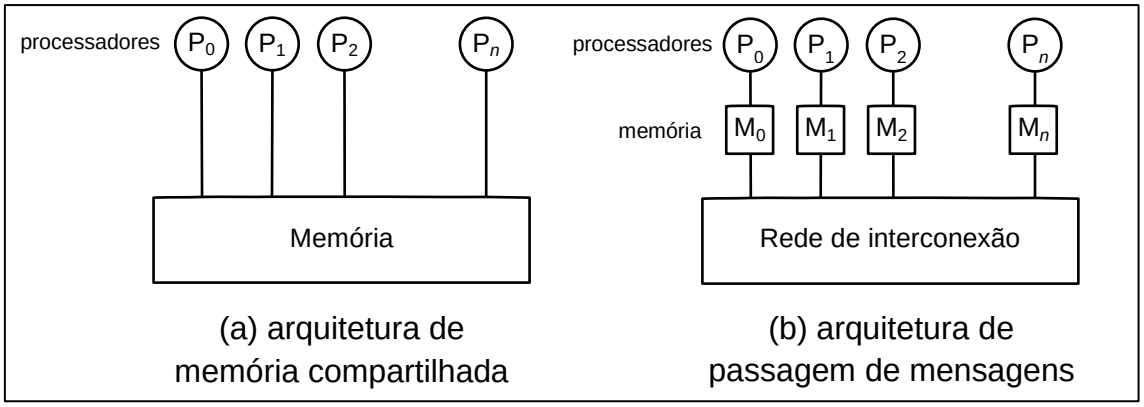


FIGURA 2 - TIPOS DE ARQUITETURAS DE COMPUTAÇÃO PARALELA (ADAPTADO DE CHANDRA *et al.*, 2001).

A OpenMP é um modelo portável e escalável que fornece memória compartilhada paralela com uma interface simples e flexível para o desenvolvimento de aplicações paralelas para as plataformas que vão desde o *desktop* até o supercomputador (OPENMP ARB, 2013). Segundo Chandra *et al.* (2001), em seu nível mais elementar, a OpenMP é um conjunto de diretivas de compilador para expressar o paralelismo de memória compartilhada. Essas diretrizes estão definidas para Fortran, C e C + + (nas linguagens C e C + +, as diretivas são referidas como "*pragmas*").

De acordo com Chapman *et al.* (2008), a *OpenMP* foi definida pelo *OpenMP Architeture Review Board* (OpenMP ARB), um grupo de fabricantes que uniu forças durante a segunda metade dos anos 90 para fornecer um meio comum para programação de computadores paralelos de memória compartilhada. Chapman *et al.* (2008) citaram também os fatores que fazem a OpenMP ser amplamente utilizada:

- ênfase na programação paralela estruturada;
- simples de usar, uma vez que toda a carga de trabalho dos detalhes do programa paralelo fica por conta do compilador;
- pode ser executada em diferentes plataformas; e

- é oportuna, pois com o forte crescimento da implantação de pequenos e grandes computadores paralelos de memória compartilhada e outros *hardwares multithreading*, a necessidade de um padrão de programação de memória compartilhada, que seja fácil de aprender e aplicar, é bem vindo em toda a indústria.

Segundo Wang *et al.* (2008), a biblioteca OpenMP fornece um modelo de execução *fork-and-join* no qual o programa inicia a execução como um *thread*, seguindo sequencialmente até uma diretiva de paralelização ser encontrada. Neste momento a *thread* cria uma série de *threads* se tornando a *thread* mestre do novo grupo. Todas são executadas até a região paralela terminar, quando são sincronizadas. Este modelo de programação é ilustrado na FIGURA 3.

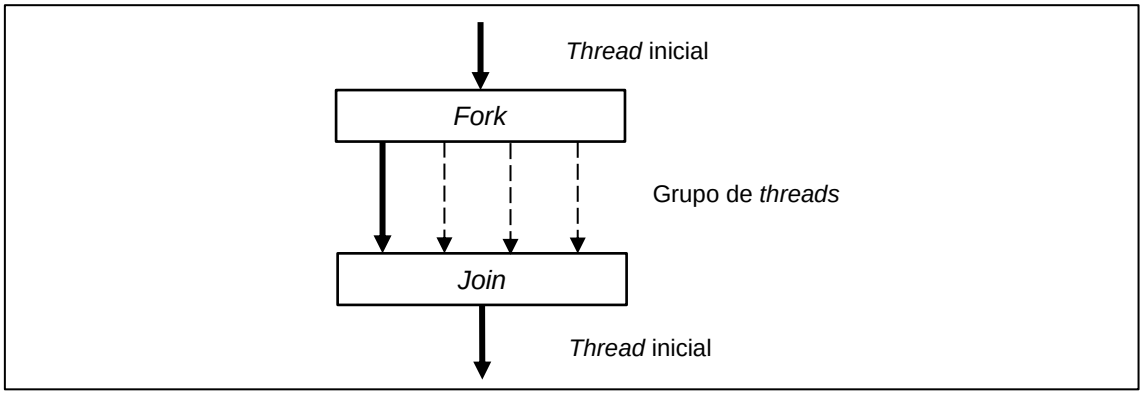


FIGURA 3 - MODELO DE PROGRAMAÇÃO *FORK-AND-JOIN* SUPORTADA NA OPENMP (ADAPTADO DE CHAPMAN *et al.*, 2008).

No caso de aplicativos paralelos em computadores que fazem uso de memória distribuída, a programação é geralmente feita com bibliotecas de passagem de mensagens explícitas como a MPI (CHANDRA *et al.*, 2001). O padrão MPI é mantido pelo Fórum MPI, um comitê formado por fornecedores, implementadores e usuários (MPI FORUM, 2014).

A MPI é amplamente utilizada na computação de ponta, onde os problemas são tão grandes que muitos computadores são necessários para resolvê-los. Ela é relativamente fácil de implementar em ampla variedade de plataformas e desta forma fornece excelente portabilidade. No entanto, a portabilidade vem a um custo. A criação de programas paralelos baseada nesta

API geralmente requer maior reorganização da sequência original do código. O esforço de desenvolvimento pode ser grande e complexo comparado a OpenMP (CHAPMAN *et al.*, 2008).

3.5.2. Implementações paralelas da PSO

Segundo Moraes (2011), apesar da reconhecida robustez do PSO em problemas de otimização, o elevado número de avaliações da função objetivo limita a sua aplicação em problemas de grande porte da engenharia. De acordo com Li *et al.* (2009), o algoritmo PSO é paralelo por natureza, visto que cada partícula calcula independentemente o seu valor de *fitness*.

O desenvolvimento de versões paralelas do algoritmo PSO é relativamente recente, quando comparado com outros métodos, como, por exemplo, o algoritmo genético e a *simulated annealing*. No entanto, a PSO paralela tem sido amplamente utilizada na última década, demonstrado o seu potencial em diversas classes de problemas, tais como a identificação de sistemas biomecânicos (SCHUTTE *et al.*, 2004 e KOH *et al.*, 2006), projeto de amortecedores magnéticos (CUI e WEILE, 2005), projeto de asa de aeronaves de transporte (VENDER e SOBIESKI-SOBIESZCZANSKI, 2005), problemas de localização não-capacitados (WANG *et al.* 2008), gerenciamento de cadeia de suprimentos (SHIMIZU e KAWAMOTO, 2008), problemas de operação de sistemas de energia (LI *et al.*, 2009 e KIM *et al.*, 2011) e otimização de estruturas treliçadas (ESPOSITO *et al.*, 2012).

Os primeiros trabalhos com versões paralelas do algoritmo PSO foram os de Schutte *et al.* (2004) e Belal e El-ghazawi (2004). No trabalho de Schutte *et al.* (2004), é proposta uma implementação síncrona do algoritmo a qual foi denominada PSPSO (*parallel syncronous particle swarm optimization*). Esses autores alertam para três questões que devem ser abordadas a fim de criar um algoritmo PSO paralelo: (1) simultaneidade de operações e escalabilidade do algoritmo; (2) implementação assíncrona *versus* implementação síncrona; e (3) coerência de implementação. Estes três temas são descritos a seguir.

Quanto à simultaneidade de operações e escalabilidade, Schutte *et al.* (2004) afirmaram que o algoritmo deve operar de tal forma que ele possa ser facilmente decomposto em operações paralelas em uma máquina com multi-processadores. A escalabilidade implica que a natureza do algoritmo não deve colocar um limite no número de nós computacionais que podem ser utilizados, permitindo assim a plena utilização dos recursos computacionais disponíveis. Segundo Moraes (2011), no caso do PSO, as duas características são satisfeitas. O cálculo da função objetivo é realizado independentemente pelas partículas do enxame podendo, portanto, ser realizado de maneira paralela. Além disso, o número máximo de nós computacionais que podem ser utilizados é uma unidade maior (no caso de utilizar a biblioteca MPI) do que o tamanho da população de partículas. Assim sendo, o limite de escalabilidade é igual ao tamanho da população.

Quanto às implementações assíncronas e síncronas, Schutte *et al.* (2004) afirmaram que o algoritmo PSO foi originalmente implementado com um esquema sincronizado para atualização do valor de *gbest*. Esta abordagem implica na realização das avaliações de fitness para todo o enxame antes de atualizar o valor de *gbest*. Experimentações posteriores revelaram que melhores taxas de convergência podem ser obtidas por meio da atualização dos valores de *pbest* e *gbest* e as suas posições após cada avaliação individual do *fitness* (ou seja, de forma assíncrona). Especula-se que devido à atualização ocorrer imediatamente após cada avaliação de fitness, o enxame reage mais rapidamente para uma melhoria no valor de *gbest*.

Em relação à coerência de implementação, Schutte *et al.* (2004) afirmaram que a paralelização não deve ter efeito adverso sobre o funcionamento do algoritmo. Cálculos sensíveis à ordem do programa devem parecer ter ocorrido exatamente na mesma ordem que na formulação síncrona em série, que conduz para exatamente a mesma resposta final. No algoritmo PSO sequencial as avaliações de *fitness* formam a maior parte do esforço computacional para a otimização e podem ser realizada de forma independente.

O algoritmo PSO, tanto o sequencial quanto paralelo, podem ser implementados de duas formas: síncrona ou assíncrona. De acordo com Esposito *et al.* (2012), na primeira, a versão síncrona, as partículas atualizam

suas velocidades e posição somente depois todas as partículas terem realizado a avaliação do valor do *fitness* de suas posições atuais. Na versão assíncrona cada partícula avalia o valor de fitness e logo em seguida atualiza sua posição. Assim, se um novo *gbest* é encontrado, este irá influenciar imediatamente as partículas não atualizadas naquela iteração.

Segundo Koh *et al.* (2006), a abordagem síncrona mantém coerência entre implementações sequenciais e paralelas, evitando assim a alteração das características de convergência do algoritmo. Assim, PSO síncrono paralelo deve obter exatamente a mesma solução final como PSO síncrona sequencial, se a sequência de números aleatórios for gerada da mesma forma pelo algoritmo tanto para a versão paralela quanto para a sequencial. Esses autores afirmaram ainda que a PSO paralela síncrona trabalha melhor quando não há heterogeneidade no tempo de avaliação do fitness e quando o número de partículas é um inteiro múltiplo do número de processadores.

Na maioria das pesquisas desenvolvidas com versões paralelas do algoritmo PSO emprega-se alguma implementação da MPI, como a MPICH (SHIMIZU e KAWAMOTO, 2008; LI *et al.*, 2009; e KIM *et al.*, 2011), a Open MPI (MORAES, 2011) ou alguma implementação da MPI não especificada (SCHUTTE et al, 2004; KOH *et al.* 2006; VENDER e SOBIESKI-SOBIESZCZANSKI, 2005). Há também um trabalho utilizando a OpenMP (WANG *et al.*, 2008) e a *Parallel Computing Toolbox* (PCT), uma biblioteca específica para processamento paralelo do MATLAB® (ESPOSITO *et al.*, 2012).

3.5.3. Implementações paralelas da BT

O algoritmo BT pode ser paralelizado de diversas formas. Talbi *et al.* (1998) classificaram os tipos de BT paralelas em duas classes básicas:

(1) *Divisão do domínio*: o paralelismo nesta classe requer alto grau de sincronização e pode ser implementado de duas formas:

a. Divisão do espaço de busca: o problema principal é dividido em subproblemas menores, cada um sendo resolvido por um algoritmo BT diferente;
b. Divisão da vizinhança: a busca pela melhor vizinhança em cada iteração é executada em paralelo, cada processo avalia um conjunto diferente da vizinhança.

(2) *Busca tabu multiprocesso*: esta classe consiste em executar múltiplos algoritmos BT em paralelo. Os processos podem ser independentes, sem nenhum tipo de comunicação entre eles, ou cooperativos, no qual, boas soluções são continuadas após determinado número de iterações e soluções ruins são reinicializadas.

Malek *et al.* (1989) implementaram a BT multiprocesso na qual o processador principal (mestre) inicia o algoritmo e, então, cria um conjunto de processos escravos que rodam o algoritmo com diferentes parâmetros (tempo tabu, lista tabu, entre outros). Depois de um determinado intervalo de tempo os processos escravos são interrompidos e o processador mestre compara os resultados. Soluções de maior qualidade continuam o processo nos processadores escravos, enquanto nos processos com soluções de baixa qualidade o algoritmo reinicia em uma nova posição com seus parâmetros zerados. Implementação semelhante foi realizada por Czapiński (2013).

Versões da BT paralela com divisão do domínio podem ser encontradas em Fiechter (1994), Czapiński e Barnes (2011) e Szymon e Dominik (2013).

Diferentemente das implementações do algoritmo PSO paralelo, na quais o emprego da biblioteca MPI é bastante comum, as pesquisas com a BT paralela empregam bibliotecas diferentes, como por exemplo, Talbi *et al.* (1998) usaram a *Parallel Virtual Machine* (PVM), Malek *et al.* (1989) utilizaram a *Parallel Program Lenguage* (PPL) uma biblioteca específica do fabricante *Sequent Computer*, Inc., e a biblioteca OCCAM foi empregada por Fiechter (1994).

Outra forma de implementar a BT paralela, ou outras meta-heurísticas, é por meio do ambiente *Compute Unified Device Architeture* (CUDA®). Segundo NVIDIA (2014), o CUDA é um plataforma de computação paralela e modelo de programação que permite aumentar o desempenho do computador

aproveitando o poder da unidade de processamento gráfico (*Graphics Processing Unit* - GPU). Para desenvolver neste ambiente faz se uso da biblioteca denominada CUDA *Toolkit*. Exemplos de implementações da BT com esta biblioteca podem ser encontradas em Czapiński e Barnes (2011), Czapiński (2013) e Szymon e Dominik (2013).

3.5.4. Avaliação de desempenho das implementações paralelas

Para medir o desempenho de programas com código paralelo, normalmente empregam-se os conceitos de *speedup* e eficiência de paralelização. De acordo com Chapman *et al.* (2008), *speedup*, é calculada pela razão entre o tempo gasto pelo programa em *P* processadores e o tempo gasto na versão sequencial, assumindo que todos os processadores requeridos estão disponíveis pela aplicação ao longo da sua execução. A eficiência de paralelização é obtida dividindo o valor de *speedup* pelo número de processadores.

Segundo Koh *et al.* (2006), idealmente, a *speedup* deve ser igual ao número de processadores e a eficiência de paralelização deve ser de 100%. No entanto, a eficiência paralela é normalmente menor devido à sobrecarga de comunicação, causada pela decomposição paralela do algoritmo de otimização e avaliações de *fitness* que exigem diferentes quantidades de tempo de computação.

3.6. *Softwares* de otimização matemática

Dentre os softwares disponíveis para a resolução de problemas de otimização tem-se aqueles proprietários, como por exemplo o LINGO, o CPLEX e o Gurobi, e os que são de código livre, tais como o GLPK, e o CBC. A seguir é apresentado breve descrição destes *softwares*.

O LINGO é uma ferramenta que permite a construção e solução de problemas de otimização matemática. Contém uma interface integrada, que inclui uma linguagem para expressar modelos de otimização, um ambiente com recursos para a construção e edição de problemas, e um conjunto de *solvers* embutidos capazes de resolver de forma eficiente a maioria das classes de modelos de otimização (LINGO, 2010).

De acordo com IBM (2014), o CPLEX *Optimization Studio* fornece uma maneira rápida de construir modelos de otimização eficientes. O CPLEX é um recurso do IBM ILOG *Optimization Studio* que oferece desempenho e robustez de última geração em um mecanismo de otimização para solucionar problemas expressos como modelos de programação matemática. O CP *Optimizer*, também um recurso do IBM ILOG *Optimization Studio*, é uma biblioteca de ferramentas de programação de restrições.

Entre os softwares proprietários um que se destaca como uma interessante opção é o *Gurobi Optimizer*, em função de este disponibilizar versão acadêmica que pode ser utilizada livremente. Segundo *Gurobi Opmization* (2014), o *Gurobi Optimizer* é um *solver* para a programação matemática. Ele inclui os seguintes *solvers*: programação linear, programação quadrática, programação quadrática restrita, programação linear inteira mista, programação quadrática inteira mista e programação quadrática inteira mista restrita.

O pacote GLPK (GNU *Linear Programming Kit*) foi desenvolvido para resolver problemas em grande escala de programação linear, programação inteira mista e outros problemas relacionados. É um conjunto de rotinas escritas em ANSI C e organizadas na forma de uma biblioteca acessível. O GLPK suporta a linguagem de modelação GNU *MathProg*, que é um subconjunto da linguagem AMPL (GLPK, 2014).

De acordo com COIN-OR (2014), o CBC (*Coin-or branch and cut*) é um *solver* de código aberto para programação inteira mista escrito em C ++. Ele pode ser usado como uma biblioteca utilizando um executável.

4. MATERIAL E MÉTODOS

4.1. Dados utilizados

O conjunto de informações para as formulações dos cenários de planejamento florestal foi obtido do banco de dados da empresa florestal REMASA Reflorestadora S.A. Estes dados constituem-se de arquivos vetoriais com as delimitações de unidades de manejo (FIGURA 4), dados cadastrais e de inventário dos talhões, bem com o manejo empregado nos talhões até o ano de 2012.

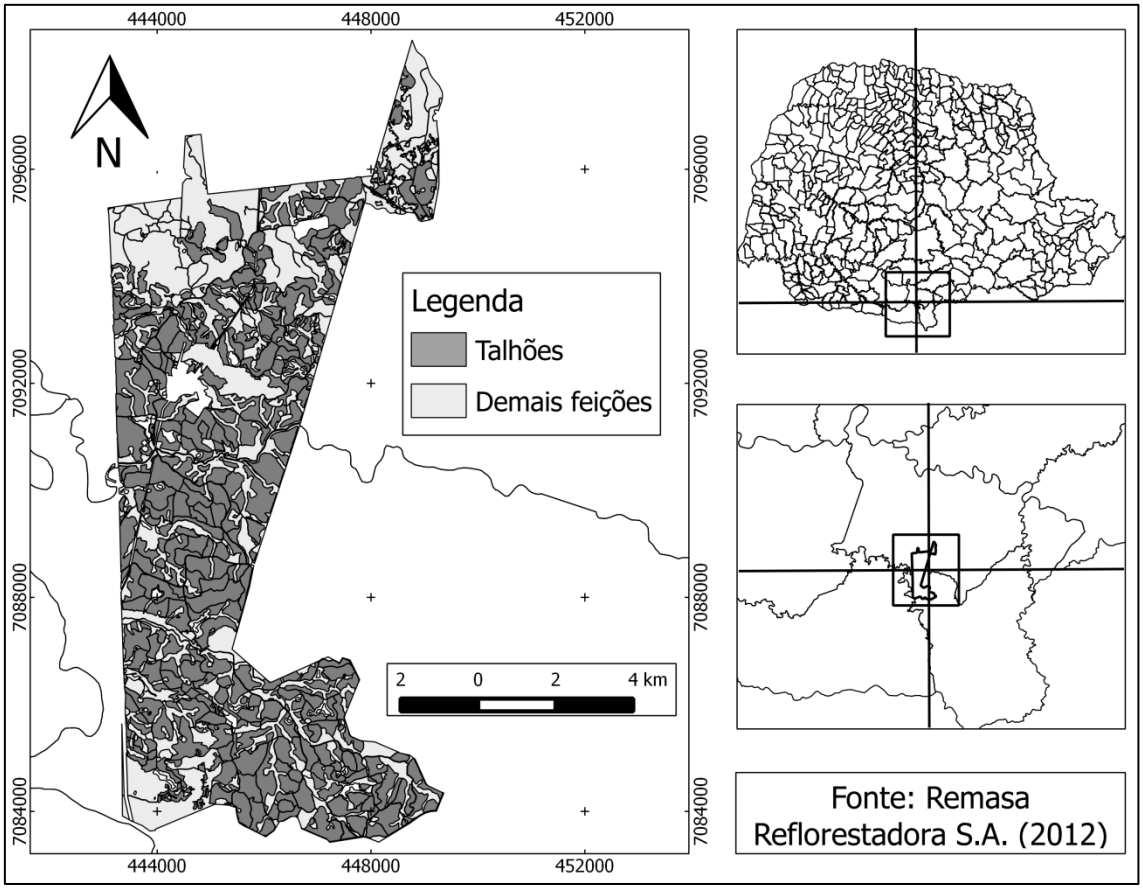


FIGURA 4 - MAPA COM A LOCALIZAÇÃO DOS TALHÕES UTILIZADOS

Para a formulação dos cenários foram empregados 236 talhões, com área total em produção de 2.429,99 hectares, os quais compõem uma fazenda

da empresa, com os gêneros *Pinus* e *Eucalyptus* (TABELA 1). As espécies de *Pinus* utilizadas pela empresa são o *P. taeda*, *P. elliottii* e *P. patula*. Quanto ao gênero *Eucalyptus* são manejadas as espécies *E. badjensis*, *E*. *benthamii* e *E*. *dunni*, os quais estão sendo gradativamente substituídos pelo gênero *Pinus* nos últimos anos devido a perdas de produção do *Eucalyptus* por geadas severas que, frequentemente, ocorrem na região.

TABELA 1 - ESPÉCIES MANEJADAS NA ÁREA DE ESTUDO.

ESPÉCIE	ÁREA (ha)	QUANTIDADE DE TALHÕES
Pinus taeda	2040,56	197
Eucalyptus spp.	227,85	18
Pinus elliottii	145,28	18
Pinus patula	16,30	3
Total	2429,99	236

A distribuição de idades dos povoamentos, considerando o ano de 2012 como base, é apresentada na FIGURA 5.

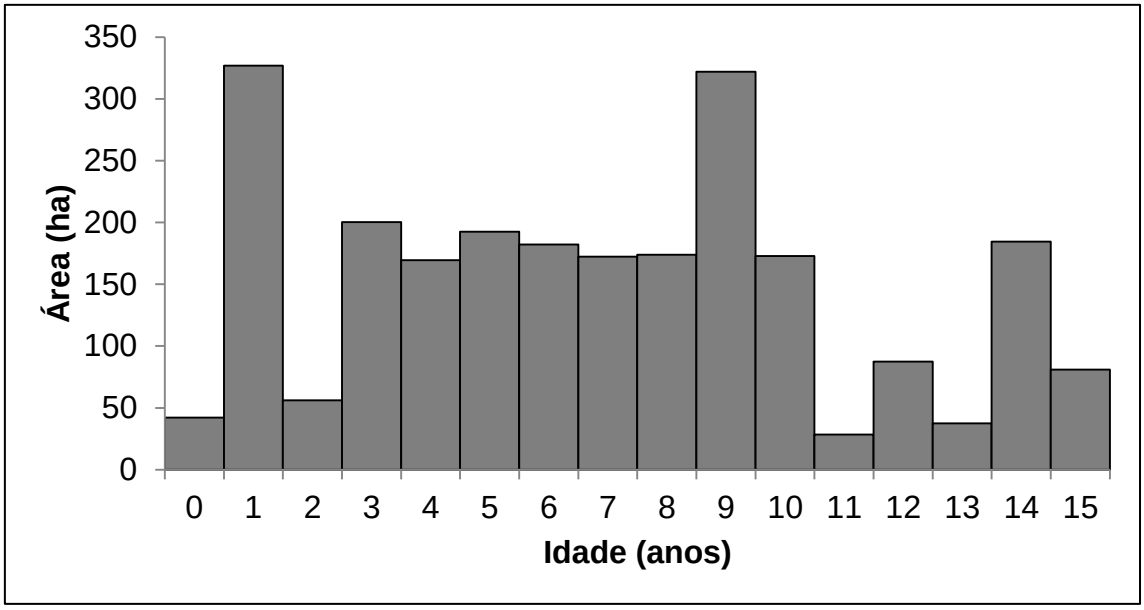


FIGURA 5 - DISTRIBUIÇÃO DE IDADE DOS POVOAMENTOS.

O espaçamento entre plantas mais utilizado na empresa é o 2,5 m x 2,5 m. Também são empregados os espaçamentos 3 x 2 m, 3 x 2,5 m e 3 x 3 m.

Na TABELA 2 são apresentadas as classes de sítio consideradas para o *Pinus taeda* e o percentual de área ocupada por cada uma destas classes.

TABELA 2 - CLASSES DE SÍTIO PARA O *Pinus taeda* NA ÁREA DE ESTUDO.

CLASSE DE SÍTIO	ÍNDICE DE SÍTIO (METROS)	ÁREA (%)
V	17	3,05
IV	19	6,14
III	21	45,09
II	23	36,19
I	25	9,52

4.2. Geração das alternativas de manejo

As alternativas de manejo (AM) de cada talhão ao longo do horizonte de planejamento (HP) foram geradas considerando o manejo do *Pinus* spp. com opções de desbaste entre as idade de 8 e 12 anos e o corte raso entre 15 e 21 anos. Para o *Eucalyptus* spp. considerou-se o desbaste aos 6 anos e corte raso entre 10 e 14 anos de idade. A opção de corte raso é seguida de plantio e a manutenção do povoamento nos anos seguintes. No caso do *Eucalyptus*, após os cortes finais o novo plantio é realizado com o gênero *Pinus*. Nos últimos anos a empresa tem substituído os plantios de *Eucalyptus* em função de perdas de produção ocasionadas por geadas severas.

Para gerar as alternativas de manejo dos cenários foi empregado o *software* OpTimber-LP (OPTIMBER, 2013), o qual permite simular, formular e resolver diversos cenários de planejamento florestal, com saídas compostas por relatórios, tabelas gráficos e mapas de fácil interpretação e compatíveis com planilhas eletrônicas (OPTIMBER, 2014).

Para as estimativas de volume de madeira para as diferentes alternativas de manejo, o OpTimber-LP utiliza os *softwares* Sispinus e SisEucalipto (OLIVEIRA, 2011). A projeção dos volumes futuros foi realizada quando havia disponibilidade de informações sobre o estado atual do povoamento, ou seja, dados provenientes de inventário. Quando estes dados não estavam disponíveis, ou no caso das estimativas do volume futuro ao longo do HP após o primeiro corte raso, foi realizada prognose do volume futuro. Os termos projeção e prognose aqui utilizados estão de acordo com as definições encontradas em DIAS (2005).

Nos casos em que foi realizado prognose da produção, o Sispinus e o SisEucalipto foram configurados com espaçamento 3 x 2 m e percentual de sobrevivência de 95%. Tanto para a projeção quanto para a prognose configurou-se nestes *softwares* o índice de homogeneidade igual a cinco.

O Optimber-LP gera alternativas de manejo considerando, na avaliação econômica, a estimativa da produção após o fim do horizonte de planejamento, até o último corte raso. Isto implica na geração de alternativas de manejo repetidas, com as mesmas opções de manejo, porém com diferentes valores econômicos. Estas alternativas de manejo repetidas podem fazer com que a quantidade de variáveis no problema aumente consideravelmente. Como o modelo utilizado neste trabalho é de programação inteira e, assim, consideravelmente sensível à quantidade de variáveis, as alternativas de manejo repetidas foram excluídas do modelo.

Nos APÊNDICES I e II são apresentados o pseudocódigo de funcionamento do algoritmo para geração de alternativas de manejo e um exemplo de funcionamento deste pseudocódigo, respectivamente.

4.3. Avaliação econômica das alternativas de manejo

Os valores de *VPL* para as diferentes alternativas de manejo foram calculados por meio da expressão (6).

$$VPL = \sum_{t=0}^{HP} R_t(1+r)^{-t} - \sum_{t=0}^{HP} C_t(1+r)^{-t} \quad (6)$$

em que: VPL: valor presente líquido do fluxo de caixa futuro de cada alternativa de manejo, em R$; C_t: custo observado no período t, em R$; R_t: receita observada no período t, em R$; r: taxa de juros; t: período do horizonte de planejamento, em anos e; HP: quantidade de períodos do horizonte de planejamento, em anos;

Para promover a equivalência temporal de valores de custos e receitas de cada alternativa de manejo utilizou-se a taxa de juros de 5 % ao ano, que é a taxa adotada pelo Banco Nacional de Desenvolvimento Econômico e Social

(BNDES) no Programa para Redução da Emissão de Gases de Efeito Estufa na Agricultura (Programa ABC), com limite de financiamento de até R$ 1 milhão por cliente e com vigência até 30 de junho de 2014 (BNDES, 2014).

As florestas em crescimento ao final do horizonte de planejamento foram avaliadas de duas formas distintas, conforme a sugestão de Schneider e Durlo (1987): (1) florestas jovens valoradas pelo valor do custo de produção e; (2) florestas maduras valoradas pelo valor comercial do estoque de madeira.

Considerou-se florestas jovens de *Pinus taeda* aquelas com idade menor ou igual a oito anos. Até esta idade as florestas com sitio III (TABELA 2) não tem estoque suficiente para que o VPL seja positivo por isso o mais indicado é valorá-las pelo custo de produção. Isto pode ser verificado na FIGURA 6 a qual foi gerada considerando a prognose da produção de uma floresta de *Pinus taeda* com índice de sítio 21, o sítio com maior representatividade na área da empresa (TABELA 2), espaçamento 3 x 2 m e sobrevivência de 95%.

No caso das florestas maduras (com idade maior que oito anos) foi utilizado uma tabela de produção gerada a partir do *software* SisPinus. Considerou-se apenas a espécie *Pinus taeda*, única que mantém crescimento ao final do HP, por conta das pressuposições do modelo explicadas no item anterior. Empregou-se o índice de sítio de 21 (classe III) espaçamento entre mudas de 3 x 2 m (1666 árvores/ha), sobrevivência de 95%, tipo de desbaste misto (sistemático e seletivo) com o corte da 5ª linha e intensidade de 50% do números de árvores. Esta tabela é apresentada no APÊNDICE III.

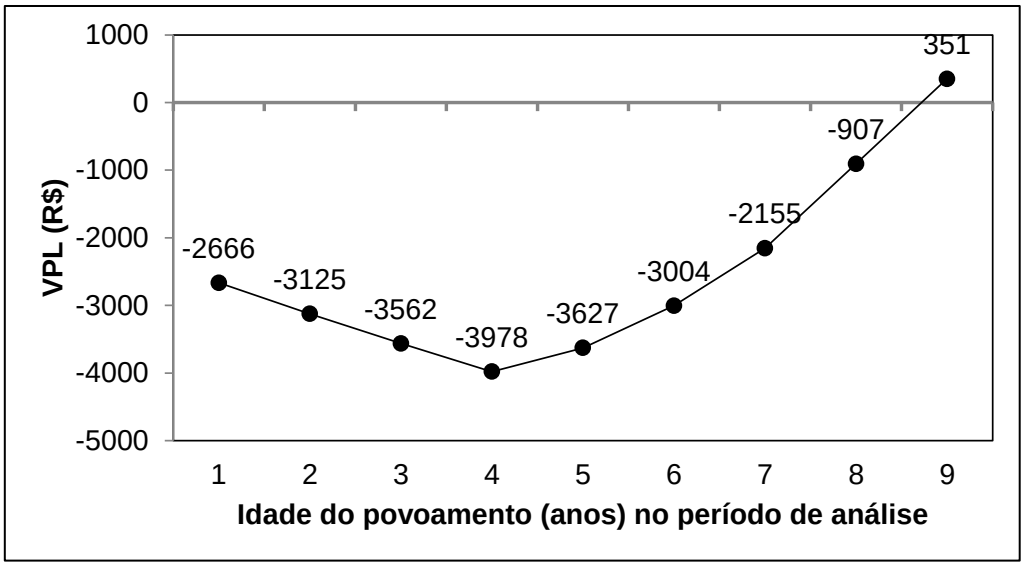


FIGURA 6 - VPL EM FUNÇÃO DA IDADE EM QUE SE REALIZA A ANÁLISE ECONÔMICA.

4.4. Preço da madeira e custo de produção

O preço de venda da madeira foi obtido do levantamento trimestral de preços florestais (PARANÁ, 2014), realizado pelo Departamento de Economia Rural (DERAL) da Secretaria de Estado da Agricultura e do Abastecimento (SEAB). Este levantamento é realizado em 21 regiões do Estado do Paraná, sendo que, para o presente trabalho, foram empregados os preços da região de União da Vitória, na qual a área da empresa está inserida. O levantamento de preços de toras é realizado em função da média dos diâmetros máximos e mínimos mais empregados no Estado fazendo diferenciação entre a tora de árvores em pé e entregues no pátio da empresa compradora. Em função da dificuldade em se obter dados relativos ao custo de colheita e transporte foi empregado o preço da tora de árvores em pé (TABELA 3).

TABELA 3 - PREÇOS DE VENDA DE MADEIRA PARA ÁRVORES EM PÉ.

ESPÉCIE	DIÂMETRO DA TORA (cm)	PREÇO (R$/m³)
Eucalyptus sp	7 a 15	28,33
Eucalyptus sp	15 a 23	66,60
Eucalyptus sp	23 a 35	80,00
Eucalyptus sp	35 acima	90,00
Pinus sp	7 a 15	28,87
Pinus sp	15 a 23	57,37
Pinus sp	23 a 35	92,25
Pinus sp	35 acima	108,00

Para o custo de produção foi considerado os custos de plantio e manutenção somente para o *Pinus* spp, uma vez que, a empresa está em fase de substituição de todos os plantios do gênero *Eucalyptus*. Em função da dificuldade em se obter dados de custo de produção, foram empregados os dados do levantamento realizado pelo DERAL na região em torno do município de União da Vitória (TABELA 4), na qual os plantios da REMASA Reflorestadora S.A estão localizados. Para os cálculos dos custos do projeto não se considerou o custo da terra ou, o também denominado, custo de oportunidade da terra, bem como o custo de manutenção após o segundo ano.

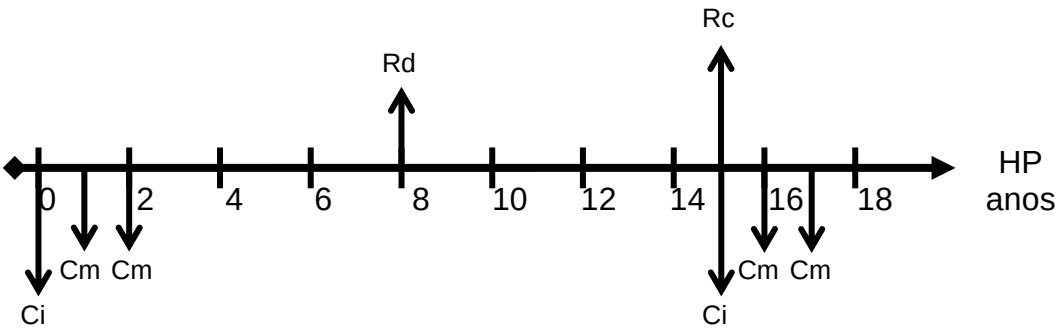
TABELA 4 - CUSTOS DE PRODUÇÃO PARA *Pinus* spp.

VARIÁVEL	UNIDADE	ANO 0		ANO 1		ANO 2	
		QUANT	R$/HA	QUANT	R$/HA	QUANT	R$/HA
Formicida	5 kg	2	187,00	2	187,00	1	93,90
Mudas	Unidade	1750*	420,00				
Herbicida	150 g	1	177,00				
Aplicação herbicida	Homem.dia	2	121,68				
Combate formigas	Homem.dia	1	60,84	1	60,84	1	60,84
Plantio	Homem.dia	4	243,36				
Coroamento	Homem.dia	5	304,20				
Roçada Manual	Homem.dia	5	304,20	5	304,20		304,20
Pós roçada	Homem.dia	6	365,04				
Custo total (R$/ha)			2184,12		552,84		458,94

* plantio 3x2m (1666 arv/ha) e replantio de 5% das mudas. FONTE: SEAB/DERAL (2013)[4]

[4] PARANÁ (Estado). Secretaria de Estado da Agricultura e do Abastecimento. Departamento de Economia Rural. **Custos de produção de florestas plantadas**. Não publicado.

Na FIGURA 7 é apresentado um exemplo de fluxo de caixa de uma alternativa de manejo considerando o desbaste aos oito anos e o corte raso aos 15 anos de idade.



Ci: custo de implantação; Cm: custo de manutenção; Rd: receita do desbaste; e Rc: receita do corte raso.

FIGURA 7 - EXEMPLO DE FLUXO DE CAIXA DE UMA ALTERNATIVA DE MANEJO.

4.5. Formulação dos modelos

O modelo utilizado foi formulado conforme o modelo clássico tipo I (JOHNSON E SCHEURMAN, 1977). Esta formulação é composta pelas expressões: (7) função objetivo que maximiza o valor presente líquido global (VPL_G) da floresta dentro de um horizonte de planejamento (HP); (8) admite somente uma alternativa de manejo para cada talhão; (9) e (10) mantém dentro de determinados limites o fluxo de volume (m³) colhido; e (11) a variável de decisão (X_{ij}) é binária, ou seja, só pode assumir valores zero ou um.

$$Maximize\ VPL_G = \sum_{i=1}^{M}\sum_{j=1}^{N} C_{ij} \cdot X_{ij} \tag{7}$$

$$\sum_{j=1}^{N} X_{ij} = 1 \qquad \forall_{i=1,2,\ldots,M} \tag{8}$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} V_{pijt} \cdot X_{ij} \geq Pmin_t \qquad t = 1,2,3,\ldots,HP \qquad p = 1,2,3,\ldots,8 \tag{9}$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} V_{pijt} \cdot X_{ij} \leq Pmax_t \qquad t = 1,2,3,\dots,HP \qquad p = 1,2,3,\dots,8 \tag{10}$$

$$X_{ij} \in \{0,1\} \text{ em que } X_{ij} = \begin{cases} 1, \text{ se a prescrição } j \text{ for adotada no talhão } i \\ 0, \text{caso contrário} \end{cases} \tag{11}$$

em que: VPL_G: valor presente líquido global da floresta, em R$; M: quantidade total de talhões; N: quantidade total de alternativas de manejo j no talhão i; C_{ij}: valor presente líquido para cada talhão i, manejado sob a alternativa j, em R$; X_{ij}: talhão i assinalado à alternativa j; V_{pijt}: volume total do produto p produzido pelo i-ésimo talhão assinalado na j-ésima alternativa de manejo, no início do período t, em R$; $Pmax_t$: produção mínima para o período t, em m³; $Pmax_t$: produção máxima para o período t, em m³; e HP: quantidade total de períodos (anos) do horizonte de planejamento.

Foram considerados oito produtos (TABELA 5) diferenciados em função do diâmetro das toras com comprimento de 2,60m. Para gerar as restrições de produção anual nos cenários, considerou-se somente o volume total dos produtos, não sendo geradas restrições para cada um dos produtos.

TABELA 5 - DIÂMETRO DAS TORAS DE CADA PRODUTO CONSIDERADO.

ESPÉCIE	DIÂMETROS DA TORA (PONTA FINA)	PRODUTO
Eucalyptus sp.	7 – 15 cm	1
Eucalyptus sp.	15 – 23 cm	2
Eucalyptus sp.	23 – 35 cm	3
Eucalyptus sp.	35 acima	4
Pinus sp.	7 – 15 cm	5
Pinus sp.	15 – 23 cm	6
Pinus sp.	23 – 35 cm	7
Pinus sp.	35 acima	8

4.6. Restrições de adjacência entre talhões

Para que os talhões fossem considerados adjacentes ou vizinhos eles deveriam compartilhar ao menos uma aresta. Talhões que compartilham

vértices não foram considerados adjacentes, conforme ilustrado na FIGURA 8. Estradas foram consideradas como arestas de ligação entre talhões.

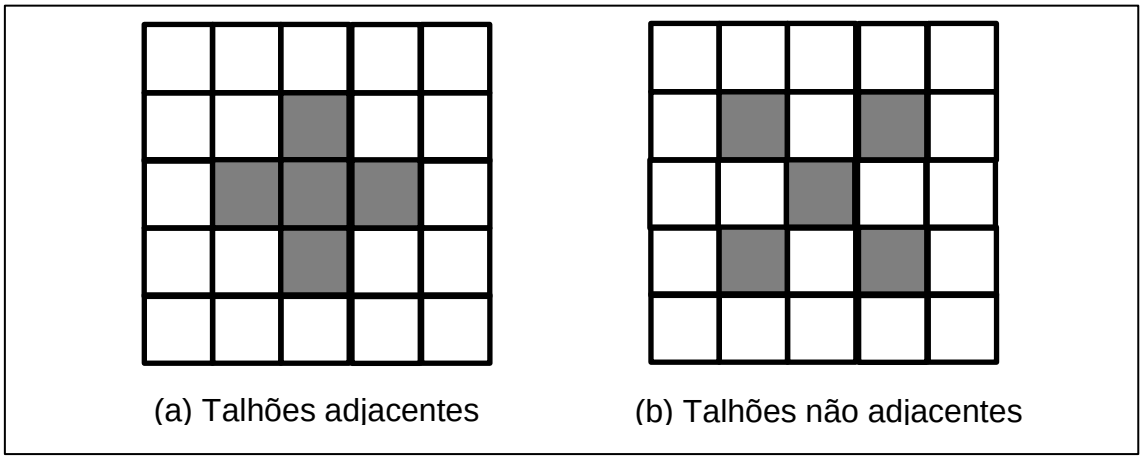


FIGURA 8 - REPRESENTAÇÃO DE ADJACÊNCIA ENTRE TALHÕES.

4.6.1. Restrição do tipo *unit restriction model* (URM)

Para a formulação da restrição do tipo URM, utilizou-se a formulação descrita por Alonso (2003). Nesta formulação, considera-se cada par de talhões adjacentes para a formulação das restrições. Desta forma, inclui-se ao modelo a expressão (12).

$$\left(\sum_{i=1}^{M} a_{ik}\right) \cdot X_{ij} + \sum_{i=1}^{M} a_{ik} \cdot X_{ij} \leq \sum_{i=1}^{M} a_{ik} \qquad \forall_{t=1,2,3,\ldots,HP} \tag{12}$$

em que: a_{ik}: matriz binária {0,1} de adjacência entre talhões, onde $i = k$.

A formulação URM utilizada no presente trabalho foi modelada conforme a descrição de McDill e Braze (2000), ou seja, a unidade de colheita é o talhão. Esta formulação pode não ser adequada quando a floresta em questão possui talhões com áreas pequenas, por exemplo, menor que 10 ha. No entanto, quando as restrições de adjacência são modeladas desta forma o modelo torna-se mais restritivo sendo um desafio para a resolução via meta-heurísticas e uma forma de avalia-las sob uma condição extrema.

4.6.2. Restrição do tipo *area restriction model* (ARM)

Para a formulação das restrições do tipo ARM foi utilizado o algoritmo *path* desenvolvido por McDill *et al.* (2002). A desvantagem deste algoritmo é que a implementação pode se tornar complexa se os *paths* forem gerados com mais de quatro talhões. Em função disto, na presente pesquisa, este procedimento foi implementado utilizando funções recursivas, as quais simplificam a implementação e garantem que *paths* de tamanhos maiores sejam gerados corretamente. O pseudocódigo deste algoritmo é apresentado no APÊNDICE IV.

A formulação das restrições ARM pode ser escrita da forma apresentada na expressão (13), conforme descrito em Gomide (2009).

$$\sum_{i \in G_i} X_{ijt} < n_{G_i} \qquad \forall_i, \forall_j, \forall_{P_i}, \forall_{t=1,2,3,\ldots,HP} \tag{13}$$

em que: G_i: conjunto de talhões adjacentes, ou grupos; e n_{G_i}: quantidade de talhões adjacentes presentes no conjunto, ou grupo.

4.7. Cenários de planejamento florestal

Avaliaram-se seis cenários para o problema de planejamento florestal com diferentes tamanhos para o horizonte de planejamento, diferentes amplitudes para a restrição de fluxo de madeira em cada período do horizonte de planejamento (HP) e dois tipos de restrições de adjacência: URM e ARM. Os cenários (TABELA 6) foram concebidos para terem dificuldade crescente. Problemas com HP mais longo resultam em espaços de busca maiores, resultando em tempo de processamento maior. Empregaram-se os cenários com 25 e 30 anos de horizonte de planejamento somente para avaliar o desempenho das versões paralelas dos algoritmos.

TABELA 6 - CENÁRIOS DE PLANEJAMENTO FLORESTAL EMPREGADOS NOS TESTES.

Nº	NOME DO CENÁRIO	TIPO DE ADJACÊNCIA	HP (ANOS)	PRODUÇÃO ANUAL (m³)		EXECUÇÕES DOS ALGORITMOS
				MÍNIMA	MÁXIMA	
(1)	URM_HP20	URM	20	40.000	180.000	100
(2)	URM_HP25	URM	25	60.000	160.000	10
(3)	URM_HP30	URM	30	80.000	140.000	5
(4)	ARM_HP20	ARM	20	80.000	140.000	50
(5)	ARM_HP25	ARM	25	90.000	130.000	5
(6)	ARM_HP30	ARM	30	80.000	140.000	3

Os valores de produção máxima e mínima foram determinados por meio de tentativa e erro. Primeiramente criou-se o cenário e tentou-se resolver via método exato. Com isso é possível descobrir se o cenário tinha solução factível. Posteriormente, com a certeza de que o cenário tinha soluções viáveis, o problema era então resolvido via meta-heurística. Testes com diferentes parâmetros são realizados nesta etapa a fim de verificar se o problema não é restritivo, a tal ponto, que as meta-heurísticas não consigam, na maioria das vezes, encontrar soluções factíveis.

4.8. Codificação das soluções

A codificação das soluções, para os dois algoritmos avaliados, foi feita de forma que em cada solução apenas uma alternativa de manejo seja atribuída a cada talhão, respeitando assim, a restrição de integridade das expressões (8) e (11). Por exemplo, considerando um cenário com sete talhões configurados com as alternativas de manejo de número 5, 13, 8, 2, 12, 15 e 6, respectivamente, para os talhões com numeração de um a sete. Este cenário pode ser representado conforme a FIGURA 9.

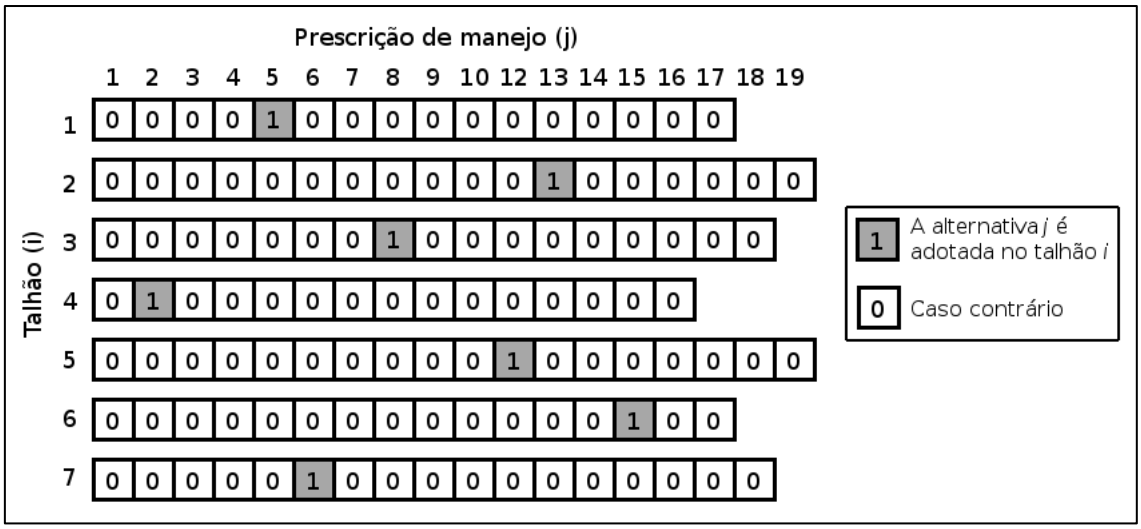


FIGURA 9 - EXEMPLO DE CODIFICAÇÃO DE UMA SOLUÇÃO.

Pode se notar na FIGURA 9 que os talhões têm quantidades diferentes de alternativas de manejo. Isto ocorre em função dos talhões terem diferentes idades e, assim, geram-se diferentes combinações de opções de manejo ao longo do horizonte de planejamento.

4.9. Algoritmo PSO sequencial

O funcionamento do algoritmo PSO sequencial foi implementada da seguinte forma: primeiro configura-se os parâmetros do algoritmo: quantidade de partículas, inércia inicial e final, coeficientes social e cognitivo, velocidade máxima e limite de iterações. Na sequência geram-se aleatoriamente as posições das partículas, limitadas pelo tamanho do espaço de busca, ou seja, a quantidade de talhões e respectivas alternativas de manejo possíveis. Depois disso, o algoritmo entra no ciclo que dura até o critério de parada ser atingido. Dentro do ciclo o algoritmo calcula a velocidade das partículas (limitada pelo valor de V_{max}), atualiza suas posições, avalia seus respectivos *fitness* e por fim atualiza os valores de *pbest* e *gbest*.

O valor da posição é arredondado para o inteiro mais próximo, porém, somente quando é realizado a avaliação do *fitness*, da mesma forma como foi implementado por Li *et al.* (2009). Isto significa que as partículas irão se

movimentar em um espaço de busca contínuo, independentemente de a posição ser um valor inteiro. Na FIGURA 10 um exemplo de como é realizado esta movimentação é apresentado, considerando um cenário bidimensional de apenas dois talhões. Depois de atualizar a posição da partícula, a qual é composta por números inteiros reais indicando o número da alternativa de manejo de cada talhão, esta é transformada em uma solução do problema, conforme a representação ilustrada na FIGURA 9.

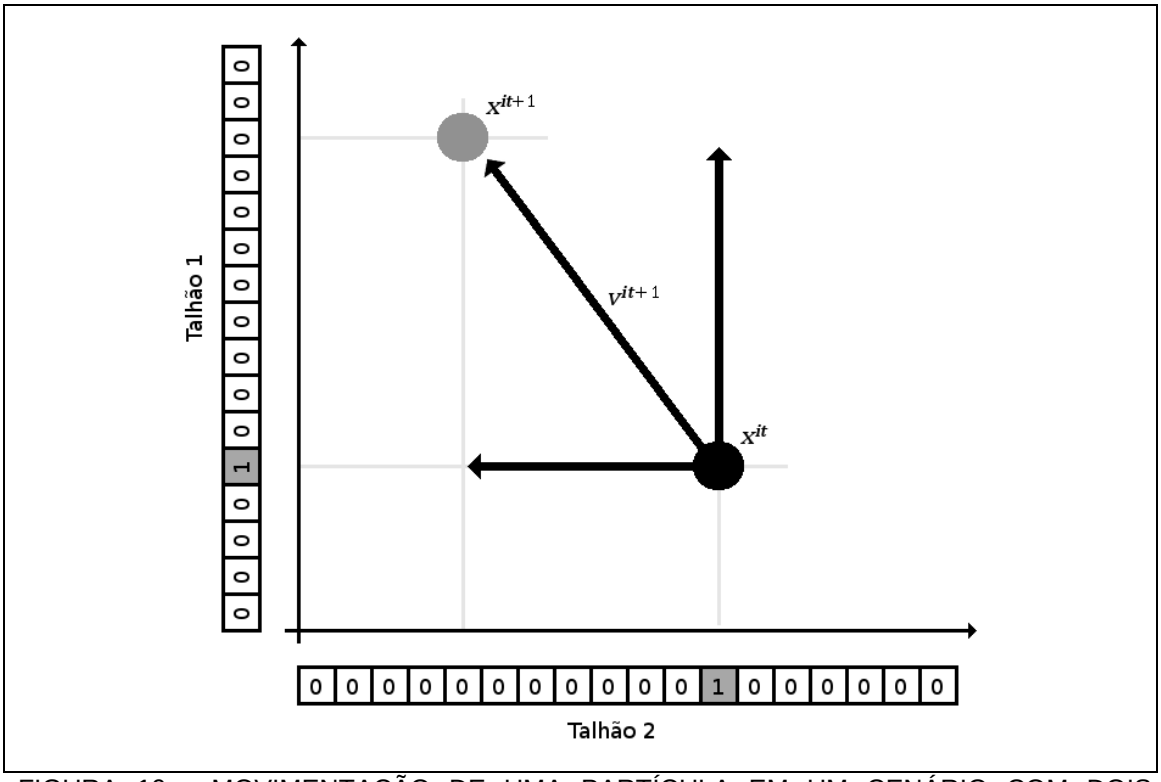


FIGURA 10 - MOVIMENTAÇÃO DE UMA PARTÍCULA EM UM CENÁRIO COM DOIS TALHÕES.

O algoritmo PSO pode ser implementado de duas formas diferentes quando se trata da atualização do *gbest*, o valor da melhor posição encontrada pelo enxame. Em uma das formas de implementação, este valor somente é atualizado após todas as partículas terem avaliado a qualidade de suas posições, ou seja, uma vez a cada iteração. Por outro lado, na segunda forma de implementação o *gbest* é atualizado imediatamente após cada partícula ter avaliado sua posição. Estes dois tipos de atualização, para o algoritmo sequencial, podem ser descritos, respectivamente, como atualização por revoada ou imediata, denominadas neste trabalho de SU-SPSO (*Swarm Update Serial PSO*) e IU-SPSO (*Immediate Update Serial PSO*), semelhante às

denominações utilizadas em Moraes (2011). A descrição do algoritmo SU-SPSO é ilustrada na FIGURA 11.

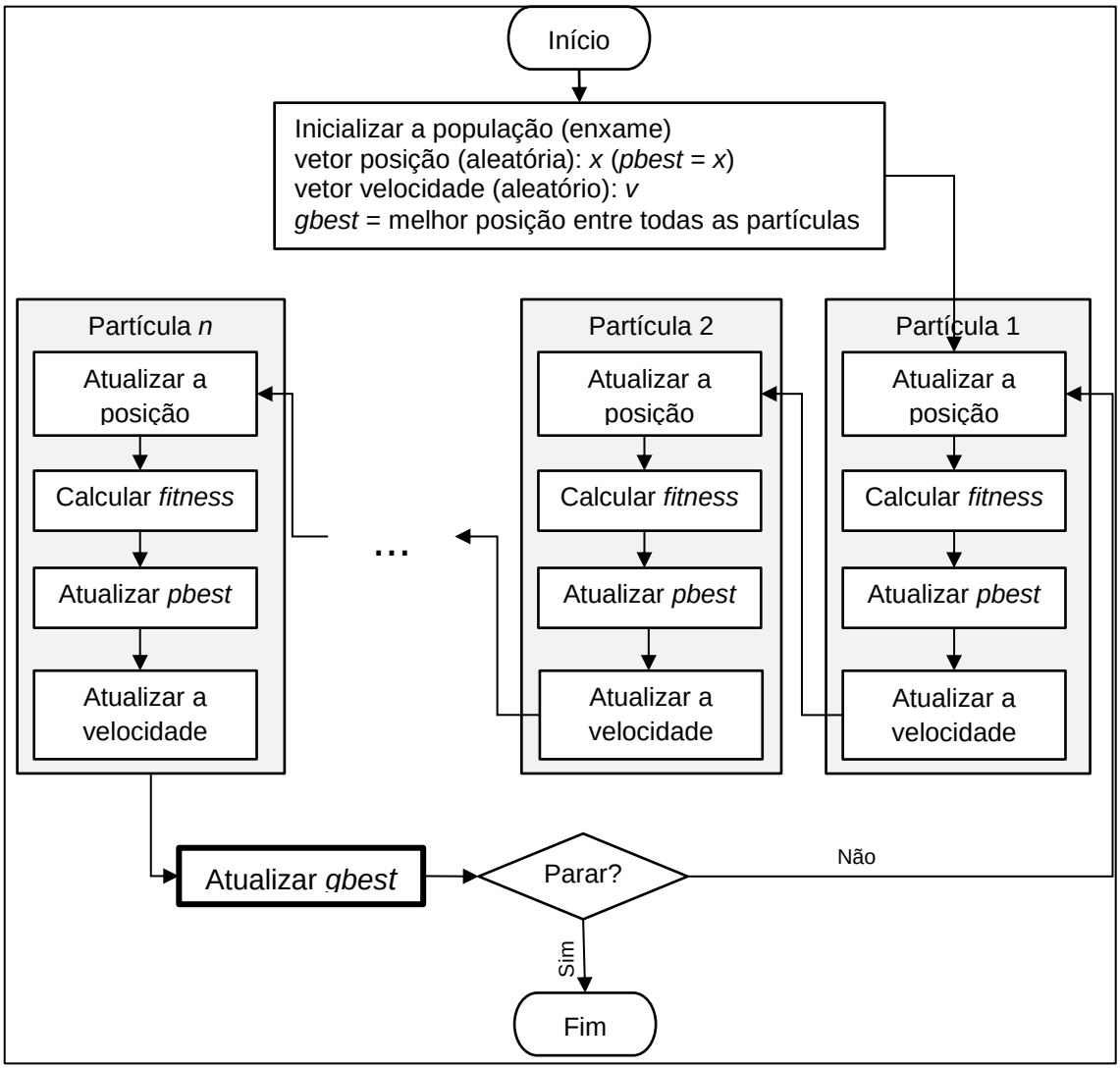


FIGURA 11 - FLUXOGRAMA DO ALGORITMO SU-SPSO.

No algoritmo IU-PSO o valor de *gbest* é imediatamente atualizado após a avaliação do *fitness* de cada partícula. O que se espera é que o enxame reaja mais rapidamente para a melhoria do valor de *gbest* e, assim, consiga convergir mais rapidamente. A descrição do algoritmo IU-SPSO é ilustrada na FIGURA 12.

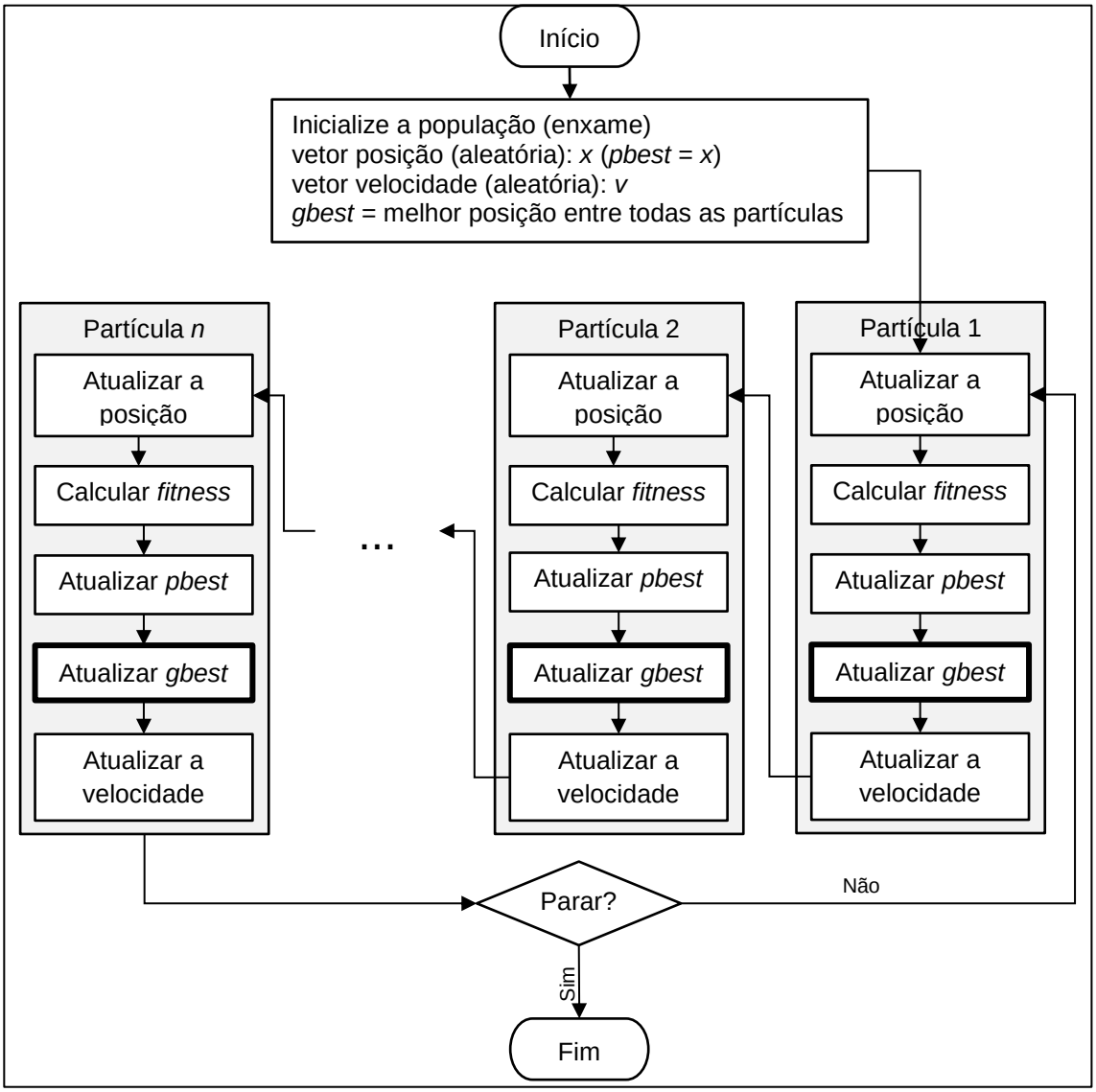


FIGURA 12 - FLUXOGRAMA DO ALGORITMO IU-SPSO.

Para a atualização da velocidade das partículas utilizou-se a expressão (14) proposta por Shi e Eberhart (1998), a qual inclui o parâmetro denominado momento de inércia (*w*).

$$v_p^{(it+1)} = w \cdot v_p^{it} + c_1 \cdot rand_1^{(it)} \cdot \left(pbest_p^{(it)} - x_p^{(it)}\right) + c_2 \cdot rand_2^{(it)} \cdot \left(gbest_p^{(it)} - x_p^{(it)}\right) \quad (14)$$

em que: x_p: posição da partícula p; v_p: velocidade da partícula p; c_1 e c_2: coeficientes cognitivos e social; $rand$: função aleatória de distribuição uniforme entre 0 e 1; $pbest_p$: melhor posição que a partícula p já obteve durante a busca;

$gbest_p$: melhor posição encontrada na vizinhança da partícula p; it: iteração atual; e w: momento de inércia.

A atualização do valor de momento de inércia a cada iteração foi feita com base na expressão (15) proposta por Eberhart e Shi (2000)

$$w_{it} = \left(w_{ini} - w_{fin}\right) \cdot \frac{(IT - it)}{IT} + w_{fin} \quad (15)$$

em que: w_{it}: momento de inércia na iteração atual; w_{ini}: momento de inércia inicial; w_{fin}: momento de inércia final; IT: número total de iterações do algoritmo; e it: iteração atual.

Para não ter que calcular o momento de inércia de cada iteração, criou-se um fator que indica a taxa de variação de *w* a cada iteração, obtido pela expressão (16).

$$w_{tx} = w_{ini} - \left(\left(w_{ini} - w_{fin}\right) \cdot \frac{(IT - 1)}{IT} + w_{fin}\right) \quad (16)$$

em que: w_{tx}: taxa de variação do momento de inércia em cada iteração.

A fórmula anterior é calculada apenas uma vez no início do algoritmo, e o momento de inércia é então atualizado a cada iteração pela expressão (17).

$$w_{it+1} = w_{it} - w_{tx} \quad (17)$$

em que: w_{it+1}: momento de inércia na próxima iteração;

4.10. Proposta de modificação ao algoritmo PSO

Ao longo de testes iniciais percebeu-se que o algoritmo PSO apresentava dificuldade em encontrar soluções factíveis, principalmente quando o problema tinha restrições de adjacência. Esta questão também foi verificada no trabalho de Shan *et al.* (2012).

Como forma de superar esta dificuldade do algoritmo PSO, foi desenvolvida uma modificação do algoritmo com intensificação de busca em vizinhança, denominada neste trabalho de *Neighbourhood Search Serial* PSO (NS-SPSO). O algoritmo proposto tem funcionamento semelhante ao método

PSO original, com uma única diferença, depois das partículas terem atualizado suas posições, uma vizinhança é gerada em torno de cada uma delas. Se alguma posição na vizinhança da partícula é melhor que a posição atual, ela se desloca para a posição vizinha. Consequentemente, os valores de *pbest* e *gbest* também são atualizados. A busca em vizinhança de uma partícula é ilustrada na FIGURA 13.

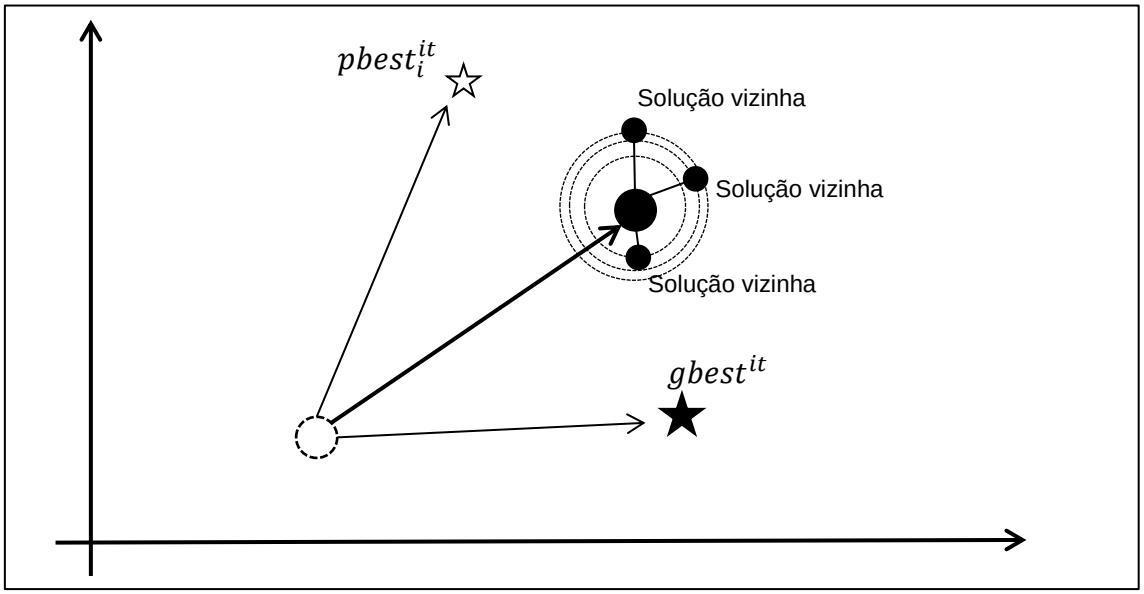


FIGURA 13 - REPRESENTAÇÃO DA BUSCA EM VIZINHANÇA DE UMA PARTÍCULA.

A forma como uma posição vizinha de uma partícula é gerada a cada iteração é igual à estratégia utilizada pelo algoritmo BT para geração de vizinhança (Item 4.12), ou seja, aleatoriamente escolhe-se um talhão e modifica-se aleatoriamente a sua alternativa de manejo.

É importante ressaltar que o algoritmo PSO proposto neste trabalho não é o algoritmo apresentado em Liu *et al.* (2007), denominado *Variable Neighbourhood Search* PSO (VNPSO), o qual combina a meta-heurística *Variable Neighbourhood Search* (VNS) e o algoritmo PSO.

4.11. Algoritmo PSO paralelo

Os algoritmos PSO paralelos desenvolvidos neste trabalho utilizam o modelo de programação denominada *fork-and-join*. Nesta estratégia, o programa inicia a execução em um único *thread* e continua até este encontrar uma região paralela do código, criando um grupo de *threads* (*fork*) e tornando-se o mestre do grupo, executando também o mesmo código dos demais membros do grupo. No fim da região paralela do código, apenas o *thread* mestre continua e os demais são encerrados (*join*).

Na FIGURA 14 é apresentado o fluxograma de funcionamento do algoritmo PSO paralelo, denominado neste trabalho de SU-PPSO (*Swarm Update Parallel* PSO). Para o desenvolvimento deste algoritmo foi utilizada as diretivas da OpenMP, particularmente a diretiva `#pragma omp parallel for`, a qual indica ao compilar que determinada região do código deve ser paralelizada. Quando não é especificada a quantidade de *threads*, a região paralela utiliza todos os que estão disponíveis. Para limitar a quantidade de threads e avaliar o desempenho do algoritmo com diferentes quantidades de processadores, foi utilizada a função OpenMP `omp_set_num_threads(int num_threads)`, a qual configura o número de *threads* que será usado pela próxima região paralela.

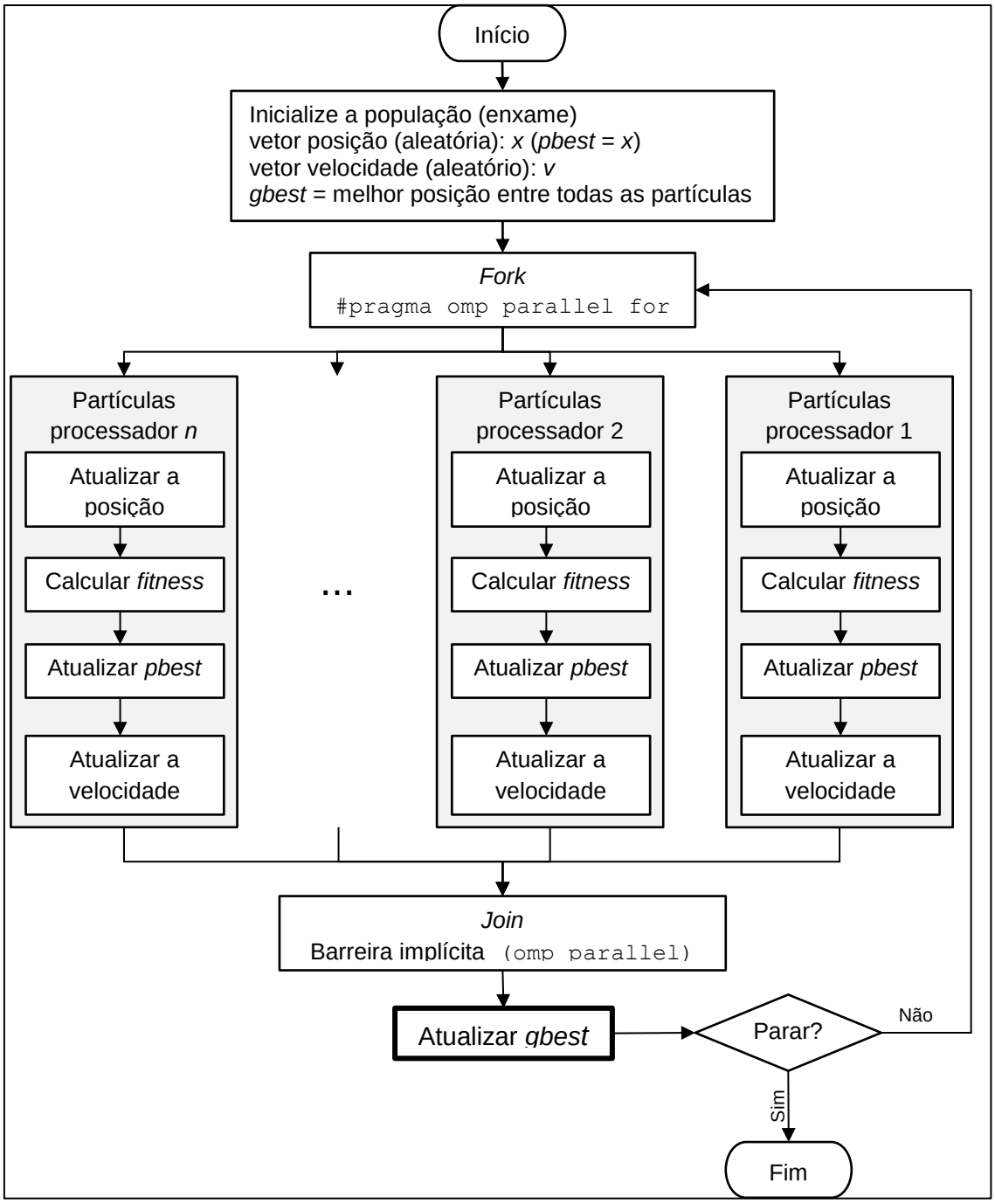


FIGURA 14 - FLUXOGRAMA DO ALGORITMO SU-PPSO.

A abordagem *fork-and-join* utilizada para o desenvolvimento da versão paralela da PSO neste trabalho resulta em um algoritmo síncrono, conforme a versão proposta por Schutte *et al.* (2004). Com isso, o algoritmo paralelo desenvolvido apresenta coerência de implementação, ou seja, a resposta final obtida pela versão paralela é exatamente igual à versão sequencial.

Também se buscou programar uma versão paralela do algoritmo com atualização imediata do *gbest*. No entanto, não se conseguiu com a OpenMP

um meio de implementar uma estratégia efetivamente mestre-escravo, que possibilitaria o desenvolvimento de um algoritmo assíncrono, conforme a versão proposta por Koh *et al.* (2006). Deste modo, a barreira implícita no final da região paralela do algoritmo faria com que o tempo de processamento fosse regulado pelo *thread* mais lento, não conseguindo assim fazer uso das vantagens de uma implementação mestre-escravo e, consequentemente, o uso máximo do recurso computacional disponível. Assim sendo, a ideia desta versão foi abandonada e as análises do algoritmo PSO paralelo focaram na versão SU-PPSO.

4.12. Algoritmo BT sequencial

O algoritmo busca tabu sequencial foi implementado utilizando a ideia de vizinhança, construída a partir de possíveis movimentos da solução atual para soluções vizinhas. Após um movimento ter sido realizado ele é classificado como tabu e armazenado na lista tabu por determinado tempo. A implementação do tempo tabu foi feito com base na quantidade de iterações.

O critério de aspiração utilizado aceita uma solução vizinha considerada tabu se esta solução apresenta uma solução melhor que a já encontrada até o momento. Por outro lado, quando todas as soluções vizinhas estão na lista tabu, utilizou-se o critério de escolher aquela com menor tempo tabu para ser a nova solução atual.

Para implementação do algoritmo BT seguiu-se quatro passos principais: (1) geração da vizinhança; (2) avaliação da vizinhança; (3) escolha da melhor solução vizinha e atualização da lista tabu; e finalmente (4), mover-se para a nova solução. O funcionamento do algoritmo BT é ilustrado na FIGURA 15.

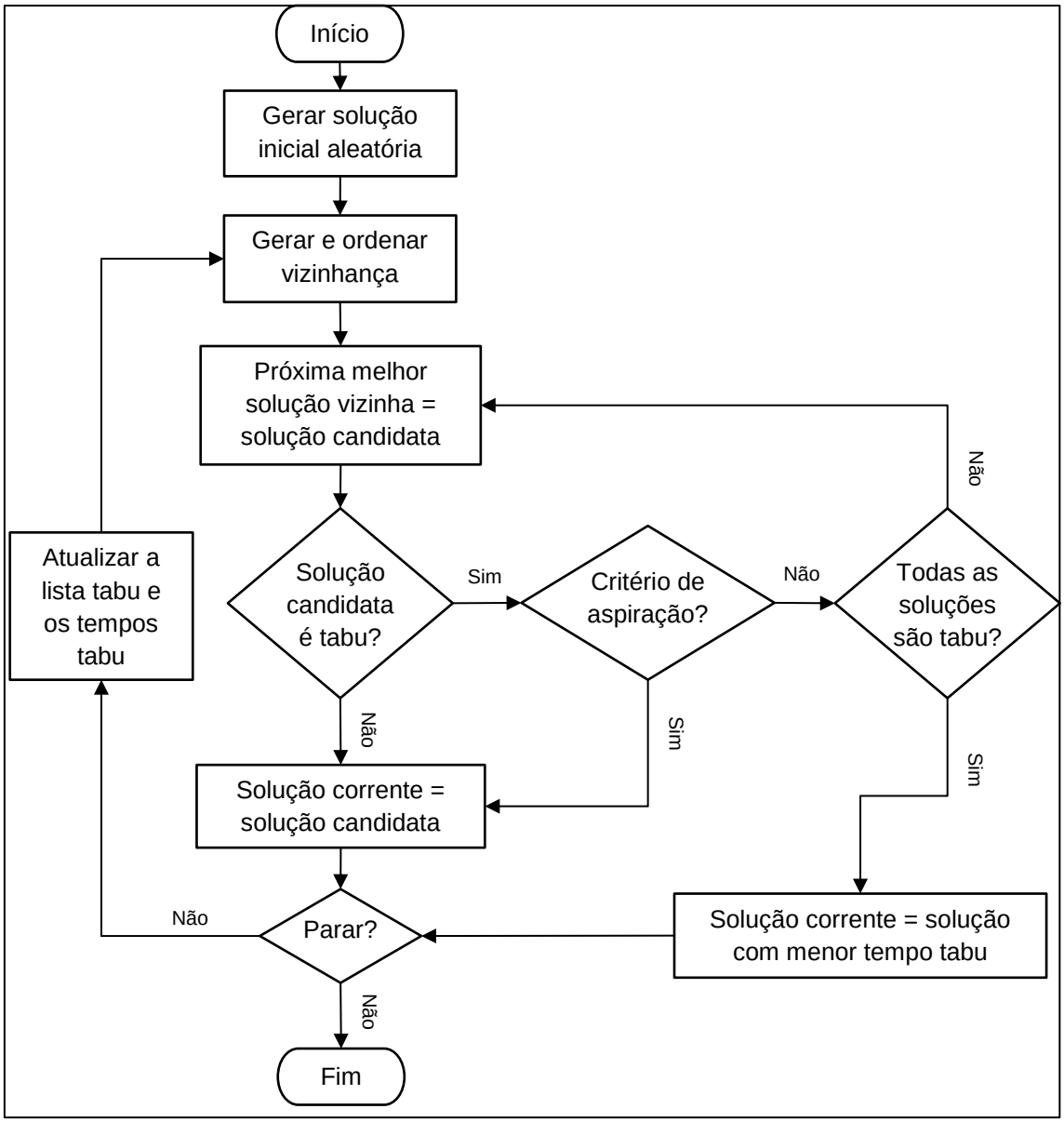


FIGURA 15 - FLUXOGRAMA DO ALGORITMO BT.

Para gerar a vizinhança de uma solução utilizou-se o seguinte mecanismo: aleatoriamente escolhe-se um talhão e modifica-se, aleatoriamente, a sua alternativa de manejo. Utilizou-se neste trabalho o conceito de vizinhança total e parcial. No caso da vizinhança total o sorteio de talhões é desnecessário, uma vez que será gerada a quantidade de soluções vizinhas igual ao número de talhões do problema. Na FIGURA 16 é ilustrada a geração de vizinhança total de uma solução com três talhões.

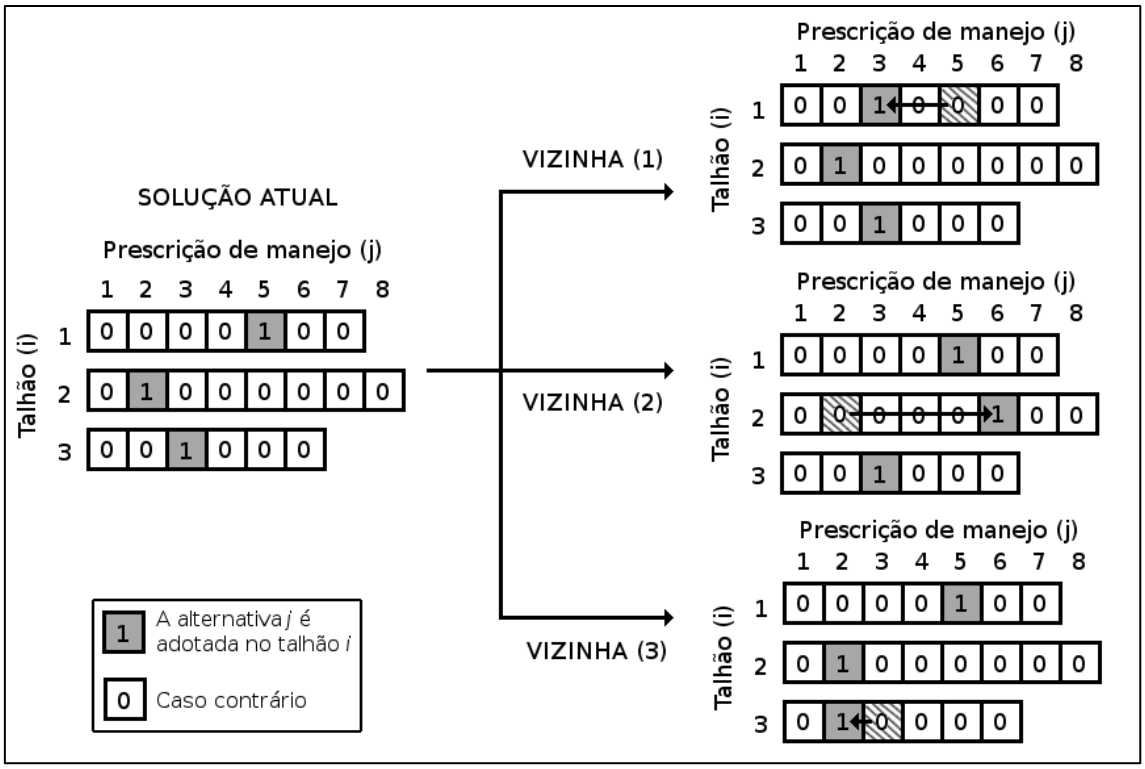


FIGURA 16 - EXEMPLO DE GERAÇÃO DE VIZINHANÇA TOTAL DO ALGORITMO BT.

4.13. Algoritmo BT paralelo

O algoritmo BT paralelo foi implementado conforme a nomenclatura “divisão de domínio” descrita em Talbi *et al.* (1998), na forma de divisão da vizinhança. Programou-se o paralelismo de forma que cada *thread* ficasse responsável pela avaliação de uma porção da vizinhança da solução atual. Na FIGURA 17 é apresentado o fluxograma de funcionamento do algoritmo BT paralelo. A parte do código que sofre modificação é após geração da vizinhança da uma solução atual, quando o processo mestre cria um grupo de *threads*, cada um com a função de calcular o *fitness* de uma parte da vizinhança gerada.

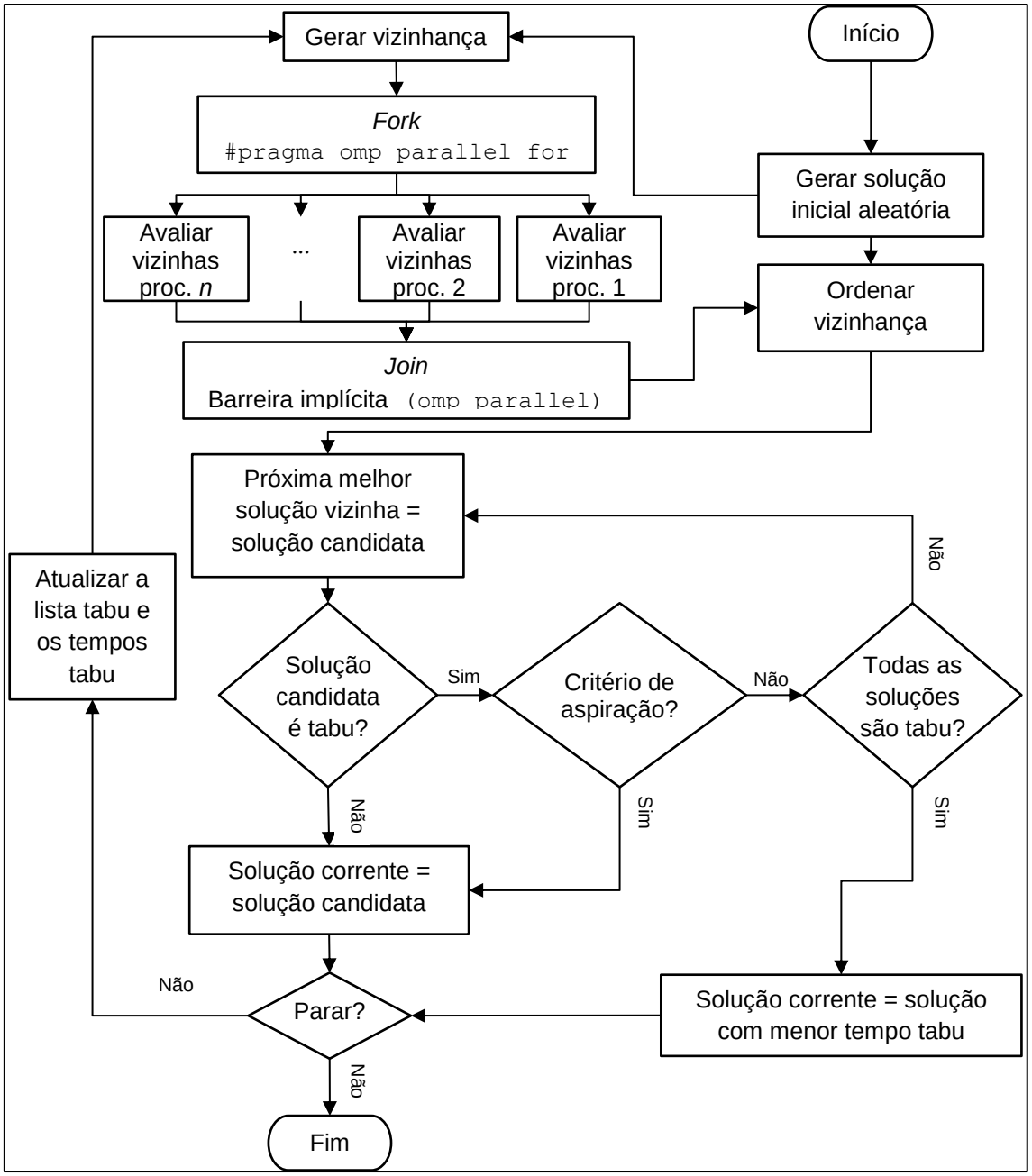


FIGURA 17 - FLUXOGRAMA DO ALGORITMO BT PARALELO.

4.14. Função de avaliação utilizada nas meta-heurísticas PSO e BT

Para medir a qualidade das soluções encontradas pelas meta-heurísticas utilizou-se o *fitness* ou aptidão da solução, no caso, a própria função-objetivo do problema de planejamento florestal subtraída de penalização que representa a violação da solução às restrições do problema.

Assim, a função-objetivo penalizada $f_p(x)$ é obtida através da modificação da função objetivo $f(x)$, conforme a expressão (18).

$$f_p(x) = f(x) - vp \cdot VT \quad (18)$$

em que: $f_p(x)$: valor da função-objetivo penalizada para a solução x; $f(x)$: valor da função-objetivo do problema; vp: penalização para cada unidade violada; e VT: violação total das restrições de produção (m³) e de adjacência.

Após os testes iniciais, o valor utilizado para a penalização (vp) foi de 80.000.000, não sendo levado em consideração se as violações ocorriam nas restrições de produção ou nas de adjacência. Este valor utilizado para o vp garante, para os cenários gerados neste trabalho, que se a solução encontrada apresentar $VT > 0$, o *fitness* será negativo ou muito abaixo do valor de uma solução factível. Assim, se ao menos dois talhões forem adjacentes em qualquer período de uma solução, esta terá seu *fitness* drasticamente reduzido incentivando os algoritmos a buscarem soluções factíveis. Por outro lado, tomou-se a precaução de que todas as soluções infactíveis fossem identificadas pelo *software* das meta-heurísticas e marcadas no relatório gerado ao final de cada execução dos algoritmos.

4.15. *Software* LINGO

Para resolução dos modelos por meio do algoritmo exato de programação inteira (*branch and bound*) foi empregado o *software* comercial LINGO, versão 13.0.2.9 *Extended* Win64, licenciado para a Universidade Federal do Paraná (UFPR). Empregou-se a configuração padrão do *software* exceto para o item *Method* da aba *Linear Solver* onde selecionou-se a opção *Barrier* e para o item *Cores to Use* selecionou-se o valor quatro e *Barrier* para em todos os *core*. O modelo no formato da linguagem do LINGO utilizada neste trabalho é apresentado no APÊNDICE V.

4.16. Parâmetros das meta-heurísticas

Buscou-se alcançar a melhor configuração de parâmetros para cada uma das meta-heurísticas testadas. Os parâmetros avaliados para a PSO estão apresentados na TABELA 7. Os demais parâmetros da PSO foram configurados da seguinte forma: w_{ini} = 0,9; w_{fin} = 0,4; c_1 = c_2 = 2; topologia de vizinhança: *gbest*.

TABELA 7 - CONFIGURAÇÃO DE PARÂMETROS AVALIADOS PARA A PSO.

Nº DE PARTÍCULAS	ITERAÇÕES	v_{max}
10	30.000	10, 20, 30, 40, 50 e 60%
30	10.000	10, 20, 30, 40, 50 e 60%
60	5.000	10, 20, 30, 40, 50 e 60%
100	3.000	10, 20, 30, 40, 50 e 60%
150	2.000	10, 20, 30, 40, 50 e 60%
200	1.500	10, 20, 30, 40, 50 e 60%
300	1.000	10, 20, 30, 40, 50 e 60%
400	750	10, 20, 30, 40, 50 e 60%
500	600	10, 20, 30, 40, 50 e 60%

O algoritmo PSO avalia a cada iteração a função objetivo do problema tantas vezes quanto for a quantidade de partículas no enxame. Por exemplo, se o algoritmo possui 50 partículas, a cada iteração o algoritmo avalia 50 novas soluções. No caso do algoritmo BT são avaliadas a cada iteração tantas soluções quanto for o tamanho da vizinhança. A fim de evitar diferenças na quantidade de avaliações do *fitness* em função das características e parâmetros de cada algoritmo, limitou-se a quantidade de avaliações em 300.000 para cada configuração dos algoritmos, independente da quantidade de iterações. Na TABELA 8 estão apresentados os parâmetros do algoritmo BT.

TABELA 8 - CONFIGURAÇÃO DE PARÂMETROS AVALIADOS PARA A BT

VIZINHANÇA		TEMPO TABU	ITERAÇÕES	AVALIAÇÕES DA FO
PERCENTUAL	ABSOLUTA			
50%	118	10, 30, 50, 70 e 100	2543	300.000
70%	165	10, 30, 50, 70 e 100	1816	300.000
100%	236	10, 30, 50, 70 e 100	1272	300.000

4.17. Avaliação das meta-heurísticas

Avaliaram-se cenários com horizontes de planejamento com períodos anuais de 20, 25 e 30 anos para os modelos tipo URM e ARM. No caso do modelo baseado na área foi empregada a área máxima de 50 ha, o qual é o valor de área máxima empregada por Gomide (2009) e próximo ao valor de 48,56 ha, ou 120 acres, que é utilizado no Estado de Oregon nos EUA (BETTINGER e BOSTON, 1999,).

Efetuou-se execuções independentes de cada abordagem das meta-heurísticas avaliadas, para cada configuração de parâmetro, para cada cenário. Realizaram-se as seguintes avaliações:

1) Desempenho dos algoritmos: comparou-se a média, desvio padrão, coeficiente de variação e valor máximo e mínimo do VPL das soluções obtidas.

2) Eficácia: mede o quão próximo a solução encontrada pela meta-heurística se aproximou da solução obtida por meio de algoritmo exato, calculada conforme a expressão (19) empregada por Rodrigues *et al.* (2003). Neste caso, devido ao elevado tempo de processamento para alcançar a solução ótima por meio do método exato, considerou-se a solução obtida pelo algoritmo *branch and bound,* após determinado tempo de execução, como a solução ótima do problema.

$$Ef = \frac{f_{meta-heurística}}{f_0} \cdot 100 \qquad (19)$$

em que: Ef: eficácia (%);$f_{meta-heurística}$: valor da melhor solução obtida em um conjunto de execuções de uma meta-heurística; f_0: valor da solução obtida pelo algoritmo exato;

3) Análise estatística: Devido à impossibilidade de provar as pressuposições para se utilizar as estatísticas paramétricas (ANOVA e teste Tukey, por exemplo), tais como a homogeneidade das variâncias e normalidade dos dados, empregou-se testes da estatística não-paramétrica. Para o teste de análise de variância não-paramétrico foi utilizado o teste de Kruskal-Wallis empregando para isso o *software* R (R CORE TEAM, 2013). O comando *kruskal.test* no R executa um teste de Kruskal-Wallis com a hipótese de nulidade de que os parâmetros de localização da distribuição são os mesmos em cada grupo ou amostra, sendo que a hipótese alternativa é a de que ao menos um grupo é diferente (R CORE TEAM, 2013).

Para o teste de comparação múltiplas não paramétrico com diferentes tamanhos de amostras, foi utilizado o teste proposto por Dunn (1964)[5], descrito em Zar (2010). Para este teste utilizou-se o erro padrão conforme a expressão (20).

$$SE = \sqrt{\frac{NA(NA+1)}{12} \cdot \left(\frac{1}{n_A} + \frac{1}{n_B}\right)} \tag{20}$$

em que: SE: erro padrão; NA: número total de observações em todos os grupos ou amostras; n_A: número de observações do grupo A e; n_B: número de observações do grupo *B*.

Na sequência calculou-se a expressão (21) possibilitando comparar o valor de Q com os valores críticos $Q_{\alpha,k}$.

$$Q = \frac{\bar{R}_B - \bar{R}_A}{SE} \tag{21}$$

em que: Q: valor a ser comparado com os valores críticos $Q_{\alpha,k}$ disponível em Zar (2010), sendo α o nível de significância e k a quantidade de grupos; $\bar{R}$: o ranking médio, ou seja, $\bar{R}_A = R_A/n_A$; R_A: soma dos rankings do grupo A.

[5] DUNN, O. J. Multiple Comparisons using rank sums. **Technometrics**, v. 6, p.241-252, 1964

Para os testes de Kruskal-Wallis e de Dunn utilizou-se o nível de significância de 5%.

4.18. Avaliação do desempenho dos algoritmos paralelos

Para medir o desempenho da versão paralela dos algoritmos PSO e BT empregou-se os conceitos de *speedup* e eficiência de paralelização, calculados por meio das Equações (22) e (23).

$$S_P = \frac{t_1}{t_P} \tag{22}$$

$$E_p(\%) = \frac{S_P}{P} \cdot 100 \tag{23}$$

em que: S_P: *speedup* para P processadores; $E_p(\%)$: eficiência de paralelização; P: número de processadores, t_P: tempo de execução do programa em P processadores; e t_1: tempo de execução do programa em um processador.

4.19. Implementação das meta-heurísticas e computador utilizado

Os códigos das meta-heurísticas, bem como dos demais programas computacionais auxiliares, foram implementados na linguagem C++, empregando o ambiente de desenvolvimento Code::Blocks, *software* distribuído sob a licença GPL v3.0. Realizou-se todas as execuções dos algoritmos em um computador com processador modelo i7-3610QM, com quatro núcleos, oito *threads* e velocidade do *clock* de 2,3 GHz.

5. RESULTADOS E DISCUSSÃO

5.1. Diferenças entre os cenários URM e ARM e resultados LINGO

Os cenários URM e ARM gerados a partir de uma floresta com 236 talhões apresentaram diferentes dimensões, como pode ser observado na TABELA 9.

TABELA 9 - DIMENSÕES DOS CENÁRIOS EMPREGADOS NOS TESTES

NOME DO CENÁRIO	TIPO DE ADJACÊNCIA	HP (ANOS)	QUANT. DE VARIÁVEIS X_{IJ}	QUANT. DE RESTRIÇÕES
URM_HP20	URM	20	17.466	1.546
URM_HP25	URM	25	44.075	1.800
URM_HP30	URM	30	96.055	2.501
ARM_HP20	ARM	20	17.466	10.084
ARM_HP25	ARM	25	44.075	15.680
ARM_HP30	ARM	30	96.055	20.401

A quantidade de variáveis de decisão, quando se mantém inalterado as opções manejo, é influenciada pelo comprimento do horizonte de planejamento, ou seja, quanto maior o HP maior é o número de alternativas de manejo possíveis para cada talhão, e assim, mais variáveis X_{ij} são geradas.

O número de restrições de produção anual mínima e máxima dos cenários é igual ao tamanho do HP multiplicado por dois, isto é, cada ano do HP tem uma restrição para a produção mínima e outra para a produção máxima.

O tipo de restrição de adjacência é o aspecto que mais influencia no tamanho dos cenários. Como pode ser observado na TABELA 9, a quantidade de restrições aumenta em função do tamanho do HP e do tipo de adjacência. Os cenários ARM geraram cerca de oito vezes mais restrições que os cenários URM. Nos trabalhos de McDill *et al.* (2002) e Gomide (2009) também foram observados cenários com mais restrições para o modelo ARM.

O software LINGO só foi capaz de encontrar solução para os cenários com o HP de 20 anos. Para os demais cenários o software exibiu mensagens com os códigos de erro “o gerador de modelos está sem memória suficiente” e “a memória virtual do sistema não é suficiente”. Outros *softwares* podem ser empregados para se conseguir obter soluções para os problemas com HP mais longos, tais como os que foram citados no item 3.6.

Na TABELA 10 são apresentados os resultados das soluções obtidas com o LINGO, as quais não são as soluções ótimas, apenas soluções obtidas pelo algoritmo *branch and bound* após determinado tempo de execução. Para fins de comparação com as meta-heurísticas, considerou-se estas soluções como as soluções ótimas.

TABELA 10 - RESULTADOS DO LINGO.

CENÁRIO	*VPL* (R$)	TEMPO DE EXECUÇÃO		ITERAÇÕES
		SEGUNDOS	hh:mm:ss	
URM_HP20	90.826.800,00	39.803	11:03:23	8.051.729
ARM_HP20	90.478.460,00	28.806	08:00:06	10.573.448

Na FIGURA 18 é apresentada a evolução da produção ao longo do horizonte de planejamento para o problema URM_HP20. Pode-se perceber que o algoritmo *branch and bound* tende a concentrar maior produção nos anos finais do HP. O volume total produzido por esta solução foi de 2.260.486,43 m³.

Na FIGURA 19 é apresentada a produção para a solução LINGO do problema ARM_HP_20. Para este problema foi possível diminuir a amplitude da restrição de produção máxima e mínima, em função das restrições do tipo ARM serem menos restritivas que a do tipo URM. Assim como para o problema URM, a solução *branch and bound* para o modelo ARM tende a concentrar a produção nos últimos anos do HP. O volume total produzido por esta solução foi de 2.240.646,42 m³.

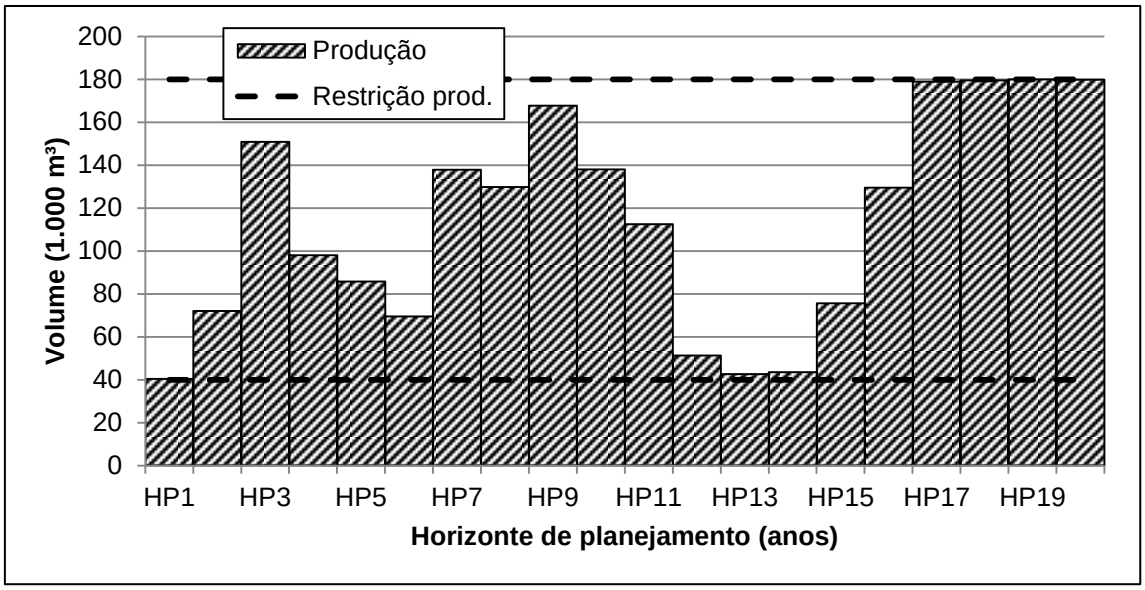


FIGURA 18 - PRODUÇÃO PARA A SOLUÇÃO LINGO DO PROBLEMA URM_HP20.

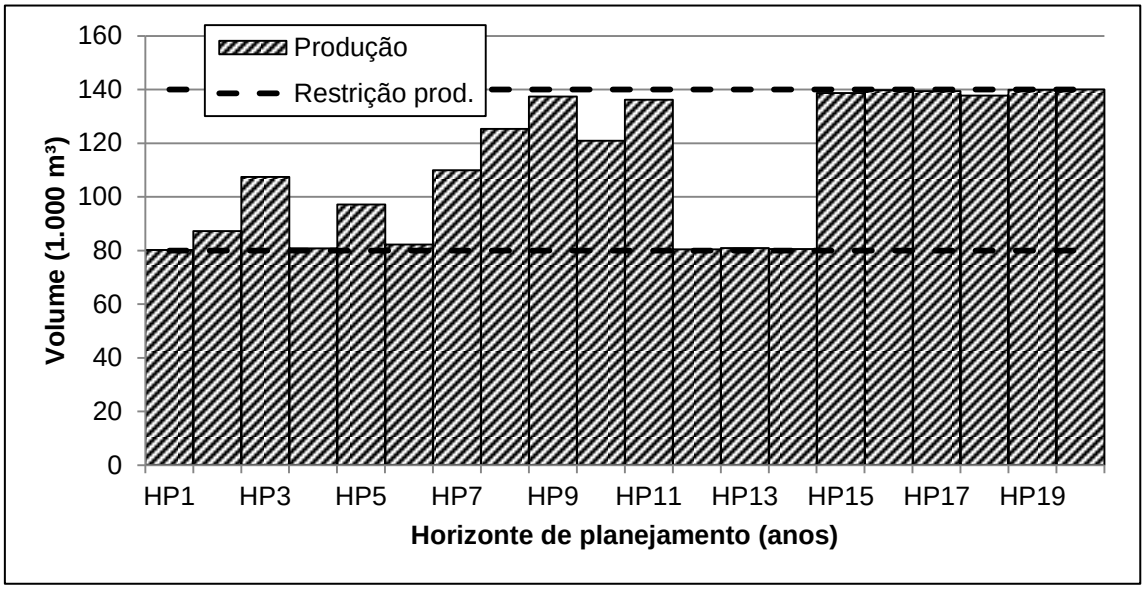


FIGURA 19 - PRODUÇÃO PARA A SOLUÇÃO LINGO DO PROBLEMA ARM_HP20.

5.2. Escolha dos parâmetros do algoritmo SU-SPSO

A melhor solução encontrada com a abordagem SU-SPSO, configurada com população de 100 partículas e V_{max} igual a 30%, para o problema URM_HP20 foi de R$ 86.522.936,00, equivalente a uma eficácia de 95,26%. Na TABELA 11 são apresentados os resultados desta abordagem, colocou-se

somente os melhores resultados, ou seja, suprimiu-se desta tabela os resultados com V_{max} maior que 30% e populações menores que 100 partículas.

TABELA 11 - RESULTADOS SU-SPSO PROBLEMA URM_HP20.

POP.	V_{max}	MÍNIMO	MÁXIMO	MÉDIA	CV%	EFICÁCIA
100	10%	85.325.048	86.241.064	85.700.342	0,22%	94,95%
100	20%	85.159.984	86.359.120	85.762.314	0,39%	95,08%
100	30%	85.050.672	86.522.936	85.649.067	0,44%	95,26%
150	10%	85.457.088	86.040.560	85.800.452	0,20%	94,73%
150	20%	85.068.392	86.172.528	85.640.026	0,32%	94,88%
150	30%	84.772.632	85.779.032	85.423.498	0,32%	94,44%
200	10%	85.307.904	86.140.632	85.806.314	0,30%	94,84%
200	20%	85.035.408	86.361.904	85.732.070	0,39%	95,08%
200	30%	84.922.264	85.847.544	85.476.996	0,32%	94,52%
300	10%	85.252.296	86.180.832	85.822.244	0,26%	94,88%
300	20%	84.817.392	86.079.224	85.666.146	0,32%	94,77%
300	30%	84.974.296	85.832.632	85.488.504	0,30%	94,50%
400	10%	85.422.944	86.266.856	85.824.129	0,25%	94,98%
400	20%	85.070.072	86.252.384	85.672.832	0,31%	94,96%
400	30%	84.761.024	85.998.088	85.398.125	0,37%	94,68%
500	10%	85.488.968	86.198.752	85.879.550	0,26%	94,90%
500	20%	85.226.888	86.125.016	85.559.475	0,27%	94,82%
500	30%	84.437.152	86.006.248	85.266.577	0,40%	94,69%

Os resultados obtidos com o algoritmo PSO, neste trabalho, apresentaram menores valores de eficácia que o algoritmo BT, conforme apresentado nos itens 5.6 e 5.7. No entanto, os resultados da PSO no presente trabalho superam os resultados apresentados em Shan *et al.* (2012), onde o maior valor de eficácia foi de 86%, mesmo o algoritmo PSO iniciando a busca em soluções factíveis.

Quando analisado a solução média da SU-SPSO verifica-se tendência do algoritmo em alcançar melhores soluções com populações maiores e valores reduzidos para o V_{max}, como pode ser observado na FIGURA 20.

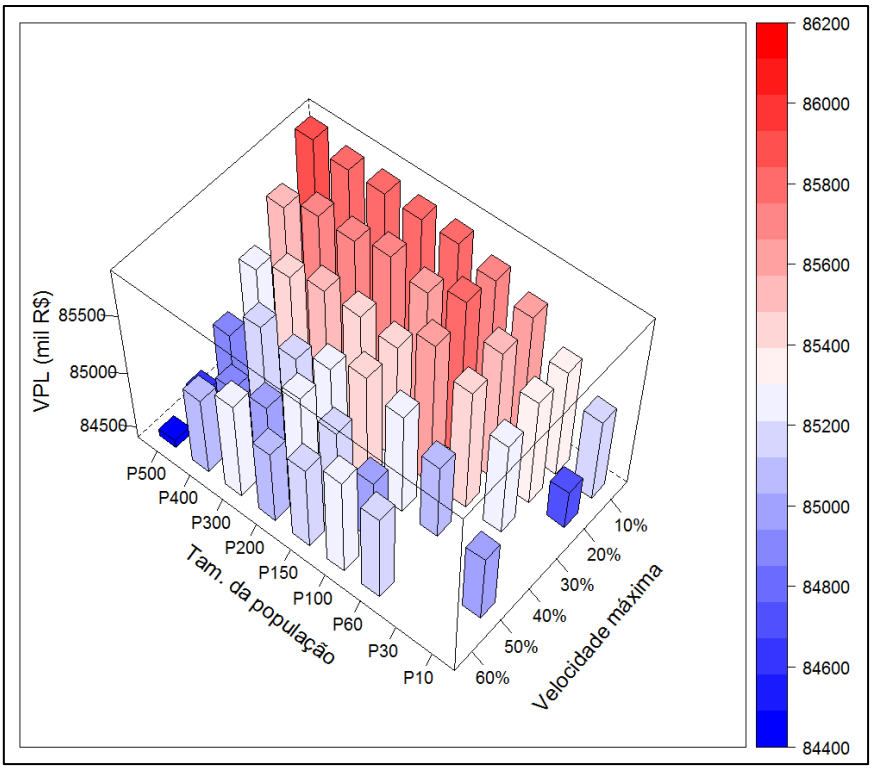


FIGURA 20 - VALORES MÉDIOS DAS SOLUÇÕES PARA A ABORDAGEM SU-SPSO PARA O PROBLEMA URM_HP20.

Na FIGURA 20 pode ser observado que quanto maior o valor do parâmetro V_{max}, menor é a média das soluções encontradas. Em relação ao tamanho da população, o algoritmo SU-SPSO é capaz de encontrar as melhores soluções com configuração a partir de 100 partículas, sendo que a partir de 150 partículas, considerando V_{max} igual a 10%, a qualidade das soluções tende a estabilizar.

5.3. Escolha dos parâmetros do algoritmo IU-SPSO

A abordagem IU-SPSO também apresentou as mesmas tendências em relação aos parâmetros de configuração população e velocidade máxima, como pode ser observado na FIGURA 21. Esta abordagem apresentou a melhor solução com população de 500 partículas e V_{max} igual a 10%. O VPL desta corresponde a R$ 86.346.808,00, equivalente a 95,07% de eficácia. No

trabalho de Brooks (2010) também encontrou-se o melhor resultado configurando a população em 500 partículas.

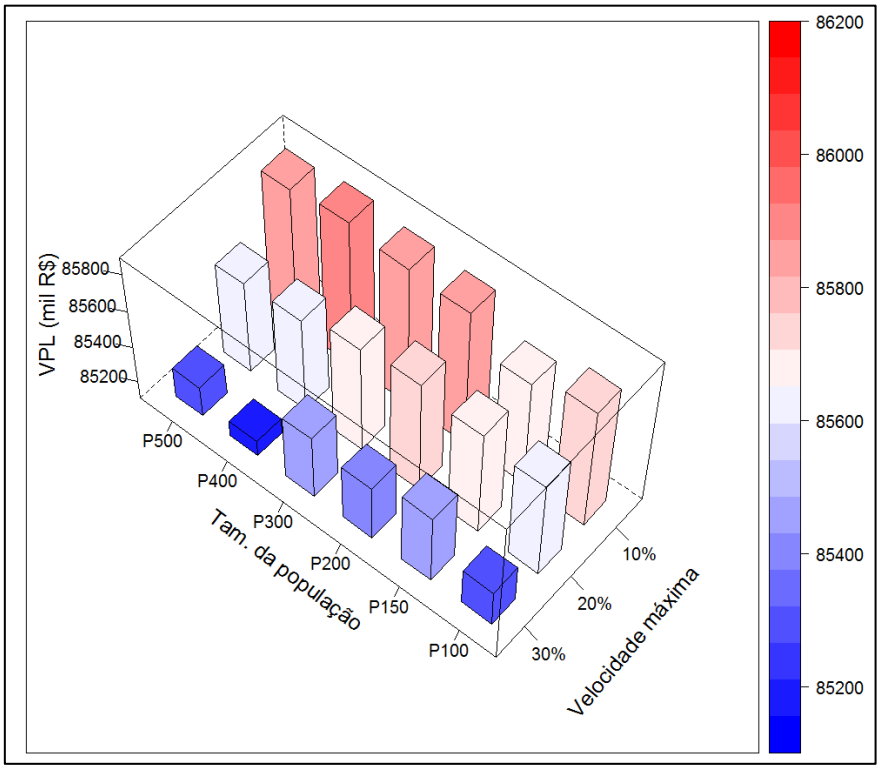


FIGURA 21 - VALORES MÉDIOS DAS SOLUÇÕES PARA A ABORDAGEM IU-SPSO PARA O PROBLEMA URM_HP20.

5.4. Comparação dos algoritmos SU-SPSO e IU-SPSO

Quando comparadas as duas abordagens, a abordagem com atualização imediata do *gbest*, IU-SPSO, apresenta, além da melhor solução, também a melhor solução média, em relação à abordagem SU-SPSO, a qual realiza a atualização do *gbest* uma vez no final de cada iteração. Esta comparação é ilustrada na FIGURA 22, onde as médias das soluções das abordagens, configuradas com V_{max} igual a 10%, são apresentadas. As soluções médias da IU-SPSO apresentam os maiores valores exceto para o tamanho de população igual a 150 e 500 partículas.

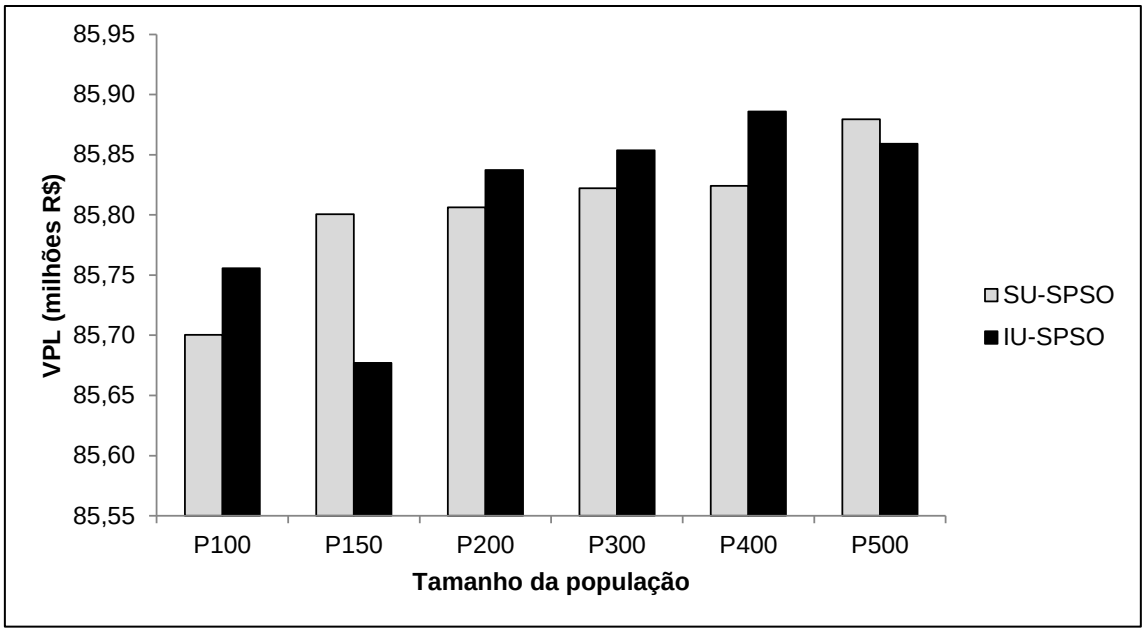


FIGURA 22 - COMPARAÇÃO DAS ABORDAGENS SU-SPSO E IU-SPSO (VALORES MÉDIOS, PROBLEMA URM_HP20, V_{max} = 10%).

A FIGURA 22 ilustra que há diferença entre os desempenhos das abordagens SU-SPSO e IU-SPSO. No entanto, é necessário determinar se esta diferença é significativa. De acordo com o teste de Kruskal-Wallis ao nível de significância de 0,05 rejeitou-se a hipótese de que todas as distribuições de soluções são iguais em todos os grupos testados. Ou seja, conforme este teste, ao menos uma das configurações de tamanho de população com as duas abordagens são diferentes. No entanto, o teste de Dunn não identificou diferenças entre as abordagens. Conforme Derrac *et al.* (2011), o teste de Dunn é muito conservador e muitas vezes não consegue detectar a diferença entre grupos. Assim, levando em conta que o teste kruskall-wallis é superior ao teste Dunn, considerou-se a abordagem IU-SPSO, que apresentou as soluções de maior valor, a versão do algoritmo a ser empregada nas demais avaliações deste trabalho.

5.5. Escolha dos parâmetros do algoritmo NS-SPSO

Nos testes iniciais do algoritmo NS-SPSO percebeu-se que os resultados não demostraram a mesma tendência de melhorar em função do

aumento do tamanho da população, como ocorreu com o algoritmo PSO original. Conforme pode ser observado na FIGURA 23, quanto o maior o tamanho da população, pior é a qualidade das soluções encontradas. Para estes testes iniciais empregou-se V_{max} igual a 10%.

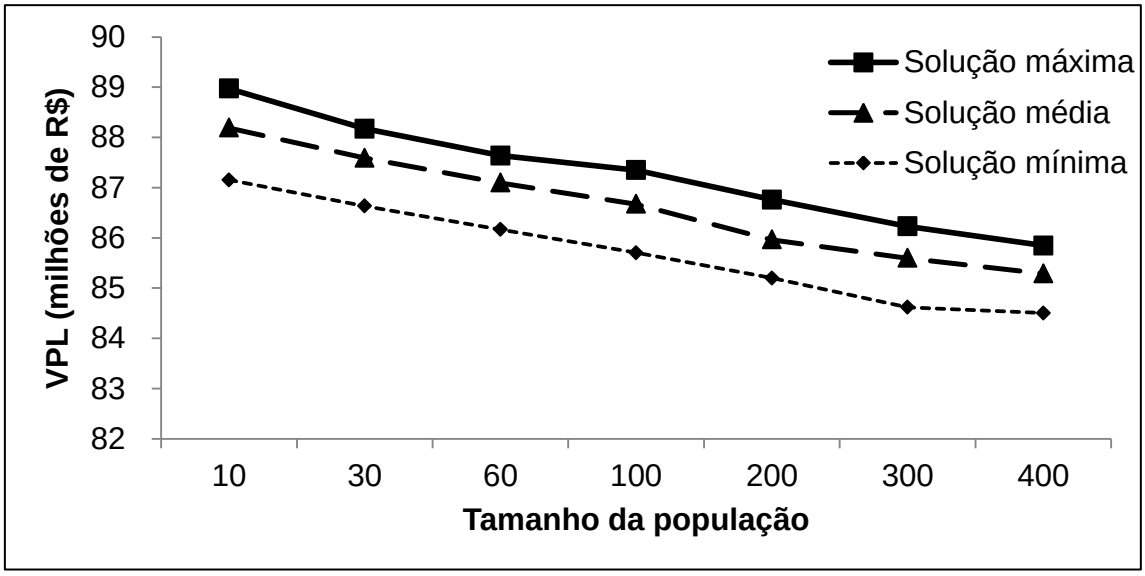


FIGURA 23 - RESULTADOS DA NS-SPSO EM FUNÇÃO DO TAMANHO DA POPULAÇÃO.

Considerando que se obtiveram os melhores resultados da NS-SPSO com a população de 10 partículas, as avaliações posteriores foram conduzidas com este tamanho de população. Na TABELA 12 são apresentados os resultados da NS-SPSO configurada com população de 10 partículas e V_{max} igual a 10%.

TABELA 12 - RESULTADOS NS-SPSO – PROBLEMA URM_HP20.

TAM. VIZINHANÇA	SOLUÇÃO FINAL – VPL (R$)			CV%	EFICÁCIA
	MÁXIMA	MÉDIA	MÍNIMA		
10	88.341.192	87.203.995	85.980.600	0,55%	97,26%
5	88.638.632	87.675.821	86.527.896	0,49%	97,59%
3	88.970.904	88.188.292	87.153.128	0,41%	97,96%
2	89.038.752	88.454.359	87.460.928	0,38%	98,03%
1	89.420.360	88.825.795	87.955.656	0,36%	98,45%

Como pode ser observado na TABELA 12, obteve-se o melhor resultado com a NS-SPSO configurada com tamanho da vizinhança igual a um. Isto pode indicar que com maiores vizinhanças em torno das posições das partículas o

algoritmo tende a ficar preso em ótimos locais. Estes resultados podem ser melhor visualizados na FIGURA 24.

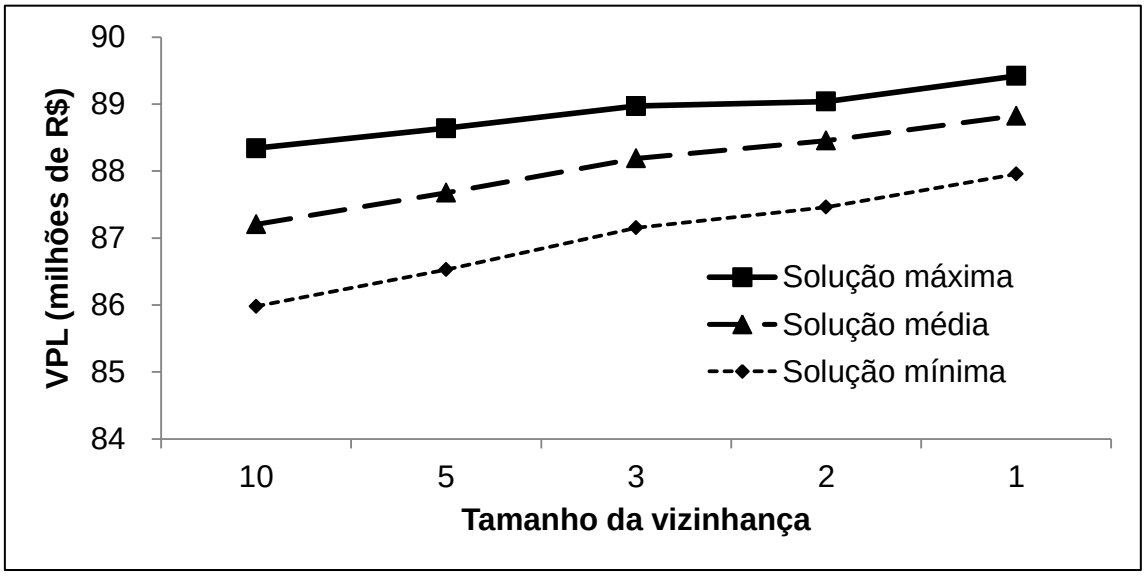


FIGURA 24 - RESULTADOS DA NS-SPSO EM FUNÇÃO DO TAMANHO DA VIZINHANÇA.

5.6. Escolha dos parâmetros do algoritmo BT

Os resultados do algoritmo BT para o problema URM_HP20 estão apresentados na TABELA 13. Conforme pode ser observado nesta tabela, obteve-se a melhor solução, a qual apresentou eficácia de 99,54%, com tempo tabu igual a 10 e vizinhança de 100%. A pior solução encontrada apresentou valor de VPL com diferença de 1,2%. Os resultados das melhores soluções estão próximos aos valores de eficácia encontrados na literatura para o algoritmo BT. Por outro lado as soluções de menor qualidade estão acima dos valores encontrados em outros trabalhos, como exemplo, pode se citar os resultados encontrados por Bettinger e Boston (1999), eficácia de 100%, Bettinger *et al.* (1999), eficácia de 99,13% e Rodrigues *et al.* (2003), eficácia de 98,84%.

TABELA 13 - RESULTADOS DO ALGORITMO BT PARA O PROBLEMA URM_HP20

TAM. VIZ.	*T*	SOLUÇÃO FINAL - VPL (R$)			CV%	EFICÁCIA
		MÍNIMO	MÁXIMO	MÉDIA		
50%	10	89.932.568	90.364.632	90.176.306	0,095%	99,49%
70%	10	89.917.760	90.355.256	90.169.788	0,108%	99,48%
100%	10	89.901.120	90.407.112	90.162.121	0,105%	99,54%
50%	30	89.955.816	90.381.544	90.183.641	0,092%	99,51%
70%	30	89.917.504	90.367.520	90.180.067	0,104%	99,49%
100%	30	89.909.200	90.382.648	90.164.577	0,105%	99,51%
50%	50	89.931.344	90.352.696	90.175.287	0,102%	99,48%
70%	50	89.910.464	90.358.152	90.183.153	0,103%	99,48%
100%	50	89.891.976	90.390.976	90.169.428	0,107%	99,52%
50%	70	89.928.704	90.336.736	90.163.063	0,102%	99,46%
70%	70	89.908.640	90.362.576	90.179.168	0,103%	99,49%
100%	70	89.900.040	90.383.576	90.172.209	0,105%	99,51%
50%	100	89.898.544	90.297.144	90.130.355	0,100%	99,42%
70%	100	89.921.768	90.329.352	90.167.796	0,107%	99,45%
100%	100	89.860.072	90.390.728	90.172.579	0,105%	99,52%
50%	150	89.737.448	90.208.944	89.983.698	0,111%	99,32%
70%	150	89.799.472	90.337.664	90.077.394	0,107%	99,46%
100%	150	89.837.968	90.328.368	90.135.089	0,104%	99,45%

Na FIGURA 25 é apresentado o gráfico com os valores médios da BT para o problema URM_HP20, onde se pode observar que o desempenho médio do algoritmo com 100% de vizinhança é o menor entre os três níveis avaliados. Isto pode ser devido ao fato de que quando o algoritmo gera mais soluções vizinhas, maior é a chance de encontrar soluções melhores que a atual. No entanto, isto provavelmente faz com que o algoritmo convirja prematuramente para um ótimo local.

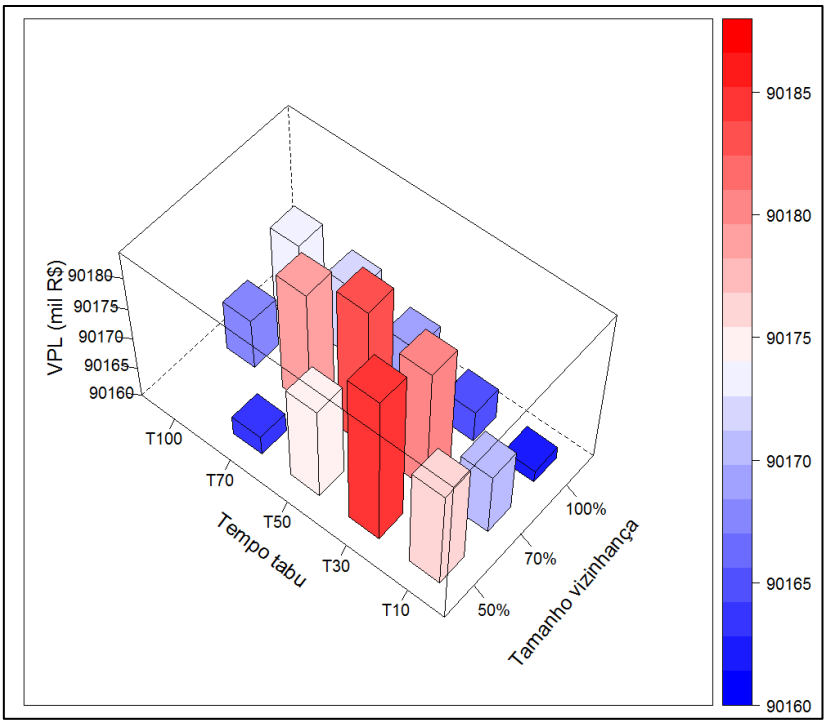


FIGURA 25 - DESEMPENHO MÉDIO DA BT - PROBLEMA URM_HP20.

Como pode ser observado na FIGURA 25 o desempenho médio da BT é melhor com as vizinhanças de 50% e 70%. No entanto, as melhores soluções são encontradas com a vizinhança de 100%, conforme pode ser verificado na FIGURA 26.

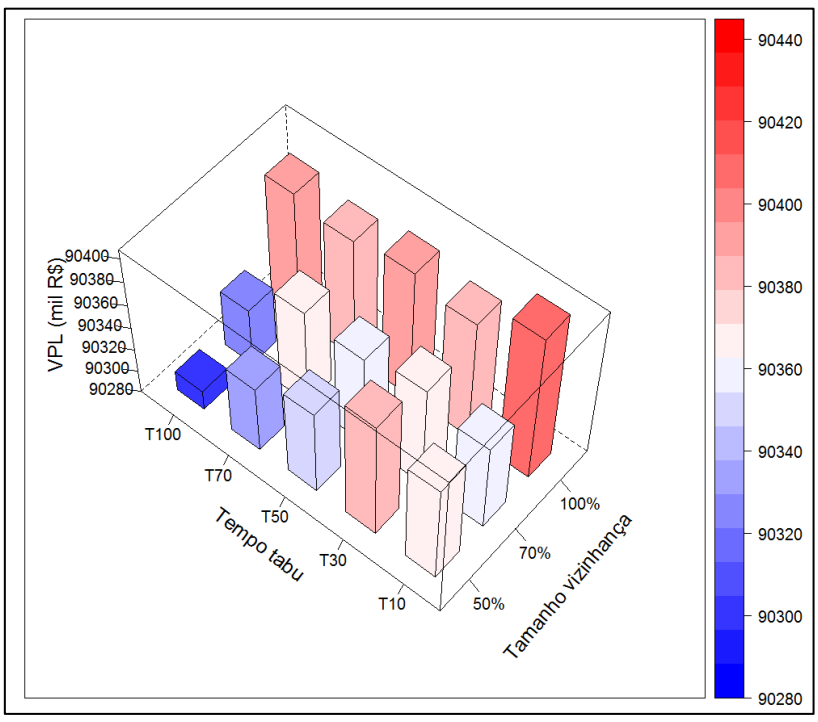


FIGURA 26 - MELHORES SOLUÇÕES DA BT - PROBLEMA URM_HP20.

5.7. Comparação das meta-heurísticas

Quando comparado à eficácia dos algoritmos avaliados, a BT apresentou os melhores resultados, tanto para o modelo URM quanto para o modelo ARM. Na TABELA 14 são apresentadas as estatísticas das três melhores soluções de cada meta-heurística para o problema URM_HP20, classificadas em função da qualidade da solução máxima.

TABELA 14 - RESULTADOS DAS META-HEURÍSTICAS - PROBLEMA URM_HP20.

META-HEURÍSTICA	SOLUÇÃO FINAL – VPL (R$)			CV(%)	EFICÁCIA
	MÍNIMO	MÁXIMO	MÉDIA		
IU-SPSO(3)	85.523.352	86.271.744	85.885.915	0,26%	94,98%
IU-SPSO(2)	85.099.336	86.346.040	85.677.671	0,33%	95,07%
IU-SPSO(1)	85.551.480	86.346.808	85.859.241	0,25%	95,07%
N-SPSO(3)	87.153.128	88.970.904	88.188.292	0,41%	97,96%
N-SPSO(2)	87.460.928	89.038.752	88.454.359	0,38%	98,03%
N-SPSO(1)	87.955.656	89.420.360	88.825.795	0,36%	98,45%
BT(3)	89.917.504	90.367.520	90.180.067	0,10%	99,49%
BT(2)	89.891.976	90.390.976	90.169.428	0,11%	99,52%
BT(1)	89.901.120	90.407.112	90.162.121	0,10%	99,54%

As soluções de melhor qualidade foram obtidas com o algoritmo BT. Além disto, a BT apresentou as menores variações entre as soluções encontradas, indicando maior robustez que os algoritmos PSO.

A proposta de modificação ao algoritmo PSO (NS-SPSO) apresentou melhores resultados que o algoritmo original, se aproximando dos resultados da BT, como pode ser observado na FIGURA 27, onde são exibidos os valores de eficácia das três melhores soluções das meta-heurísticas.

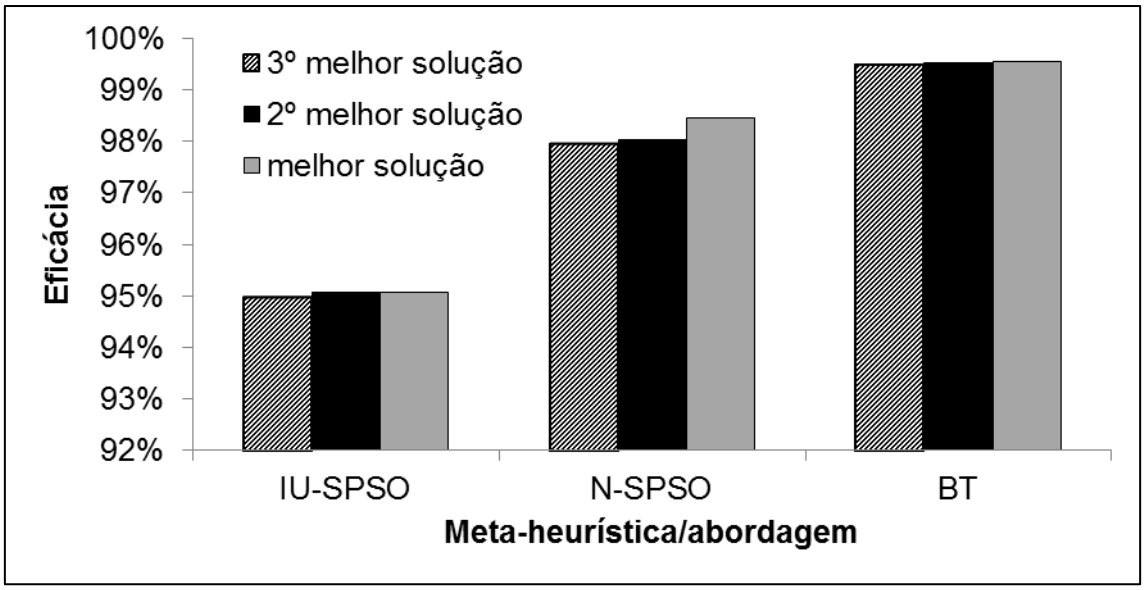


FIGURA 27 - VALORES DE EFICÁCIA DAS TRÊS MELHORES SOLUÇÕES DAS META-HEURÍSTICAS - PROBLEMA URM_HP20.

É importante ressaltar que o modelo empregado neste trabalho contém somente um objetivo, maximizar o retorno econômico, e dois tipos de restrição, de produção e de adjacência, e por isso as meta-heurísticas apresentam valores de eficácia próximos de 100%. É provável que problemas com mais objetivos e restrições apresentem maiores dificuldades para as meta-heurísticas.

Na FIGURA 28, na FIGURA 29 e na FIGURA 30 são apresentadas as produções das melhores soluções dos algoritmos IU-SPSO, N-SPSO e BT, respectivamente. Pode-se observar que as soluções das meta-heurísticas seguem a tendência da solução ótima em concentrar a produção nos últimos anos do planejamento. Os demais produtos apresentados nestas figuras são as toras de *Eucalyptus* sp dos quatro diâmetros considerados (TABELA 3) e de *Pinus* sp com diâmetro de tora acima de 35 cm. No APÊNDICE VII é apresentado um exemplo da localização dos talhões agendados para colheita em cada período do HP com a melhor solução encontrada pela BT.

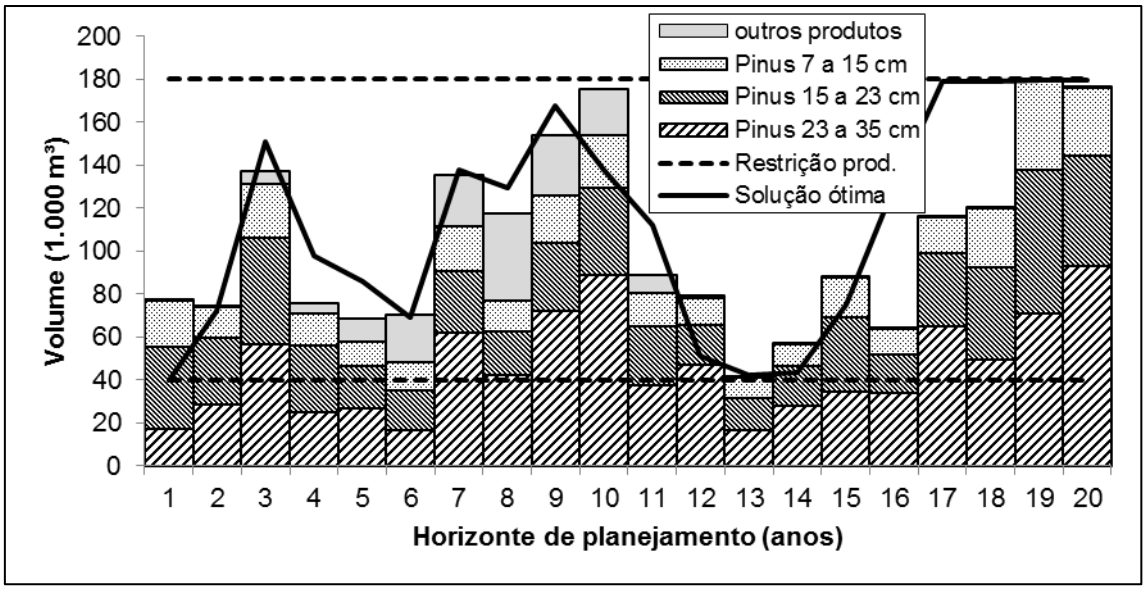


FIGURA 28 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ABORDAGEM IU-SPSO.

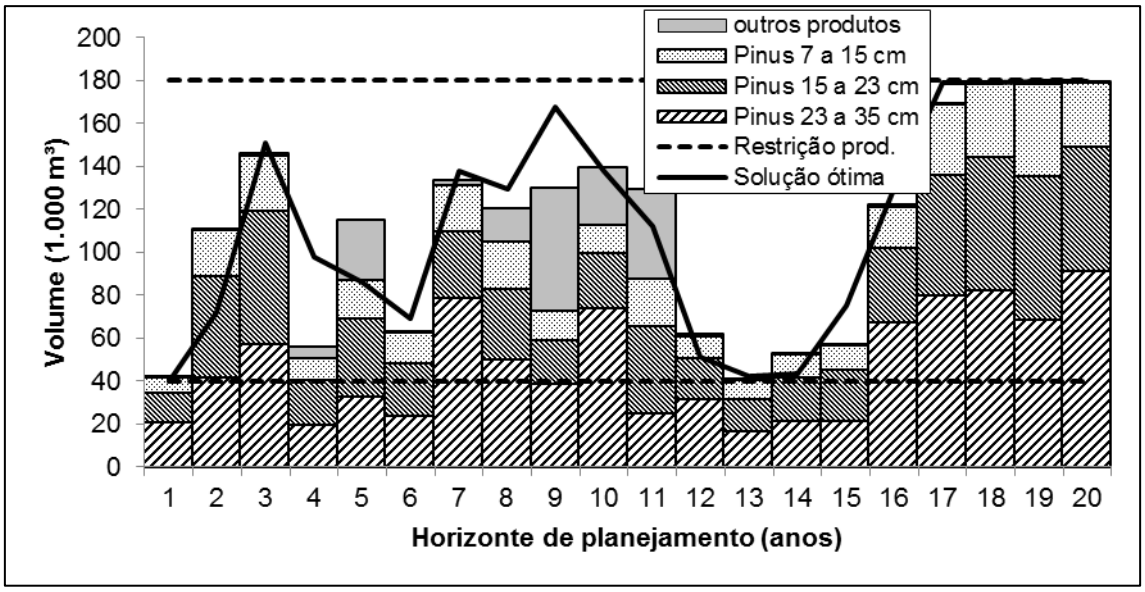


FIGURA 29 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ABORDAGEM N-SPSO.

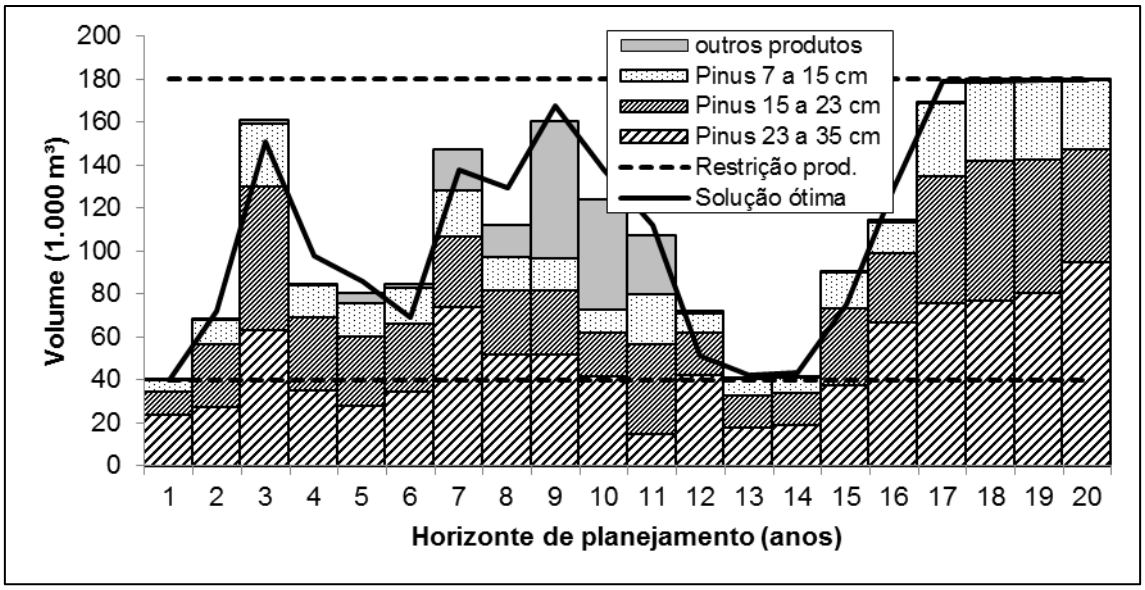


FIGURA 30 - PRODUÇÃO PARA O PROBLEMA URM_HP20, MELHOR SOLUÇÃO ALGORITMO BT.

Conforme o teste de Kruskal-Wallis ao nível de significância de 0,05 rejeitou-se a hipótese (p-valor = $5{,}87\times10^{-120}$) de que todas as distribuições de soluções são iguais em todos os grupos testados. Apesar da diferença entre as meta-heurísticas serem visualmente identificáveis na FIGURA 27, somente foi possível provar estatisticamente que a BT é diferente das duas abordagens da PSO, conforme o teste de Dunn apresentado na TABELA 15.

TABELA 15 - RESULTADO DO TESTE DE DUNN PARA O PROBLEMA URM_HP20.

META-HEURÍSTICA	n	$\bar{R}$					
BT(1)	100	536,4	**a***				
BT(2)	100	523,5	**a**				
BT(3)	100	516,6	**a**				
N-SPSO(1)	99	296,6		**b**			
N-SPSO(2)	100	219,3		**b**	**c**		
N-SPSO(3)	100	162,8			**c**	**d**	
IU-SPSO(1)	17	48,5				**d**	**e**
IU-SPSO(2)	22	45,0				**d**	**e**
IU-SPSO(3)	37	30,0					**e**

* grupos seguidos de mesma letra não apresentam diferença significativa de acordo com o teste de Dunn ao nível de significância de 0,05

Na TABELA 16 são apresentados os três melhores resultados das meta-heurísticas para o problema ARM_HP20. Novamente a BT apresentou as

melhores soluções, embora para esta configuração do problema tenha apresentado CV% próximo às duas abordagens da PSO.

TABELA 16 - RESULTADOS DAS META-HEURÍSTICAS - PROBLEMA ARM_HP20.

META-HEURÍSTICA	SOLUÇÃO FINAL – VPL (R$)			CV(%)	EFICÁCIA
	MÍNIMO	MÁXIMO	MÉDIA		
IU-SPSO(3)	85.424.304	86.455.720	86.092.231	0,25%	95,55%
IU-SPSO(2)	85.678.616	86.536.064	86.104.646	0,29%	95,64%
IU-SPSO(1)	85.425.368	86.559.776	86.001.687	0,32%	95,67%
N-SPSO(3)	87.081.080	88.195.896	87.648.660	0,29%	97,48%
N-SPSO(2)	87.250.296	88.534.976	87.846.163	0,33%	97,85%
N-SPSO(1)	87.356.064	88.587.728	88.098.410	0,32%	97,91%
BT(3)	88.759.984	89.519.408	89.155.824	0,24%	98,94%
BT(2)	88.793.480	89.716.608	89.208.384	0,27%	99,16%
BT(1)	88.792.176	89.699.976	89.238.316	0,26%	99,14%

Na FIGURA 31 são ilustrados os valores de eficácia das melhores soluções dos algoritmos avaliados.

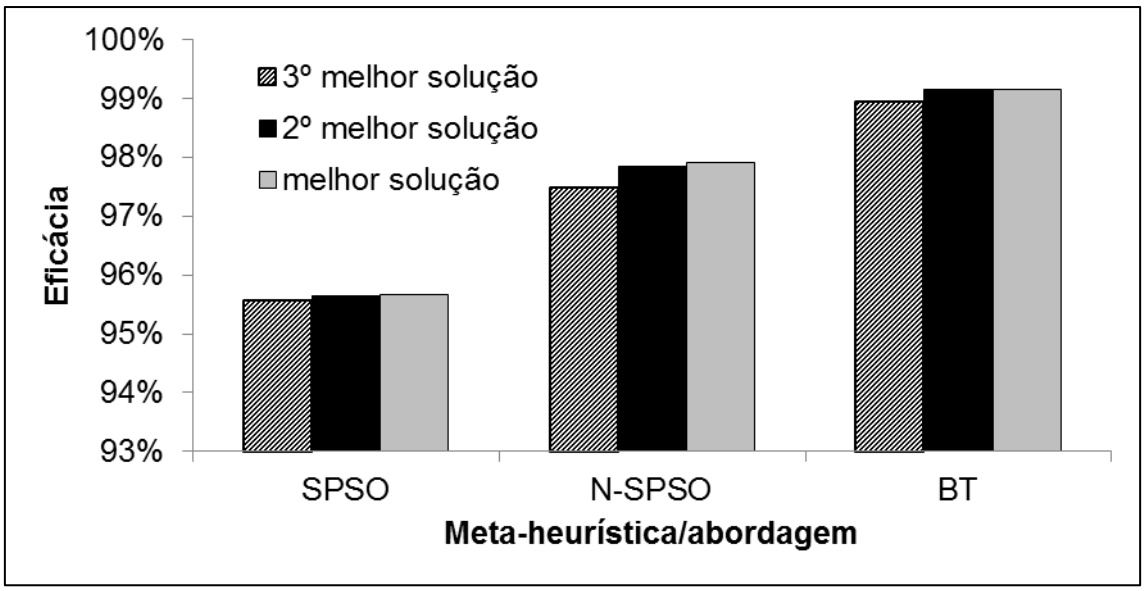


FIGURA 31 - VALORES DE EFICÁCIA DAS TRÊS MELHORES SOLUÇÕES DAS META-HEURÍSTICAS - PROBLEMA ARM_HP20.

Na FIGURA 32, na FIGURA 33 e na FIGURA 34 são apresentadas as produções para as melhores soluções das meta-heurísticas testadas para o problema ARM_HP20. Com o modelo do tipo ARM foi possível diminuir a amplitude da restrição de produção, tornando a solução encontrada mais

interessante do ponto de vista operacional. Por outro lado, o modelo URM, por ser mais restritivo, foi configurado com valores mais relaxados para a restrição de produção anual, uma vez que as meta-heurísticas apresentaram dificuldade em encontrar soluções factíveis para menores variações entre produção máxima e mínima. No APÊNDICE VI é apresentada um exemplo de agendamento de colheita para todos os talhões para a melhor solução encontrada com a BT.

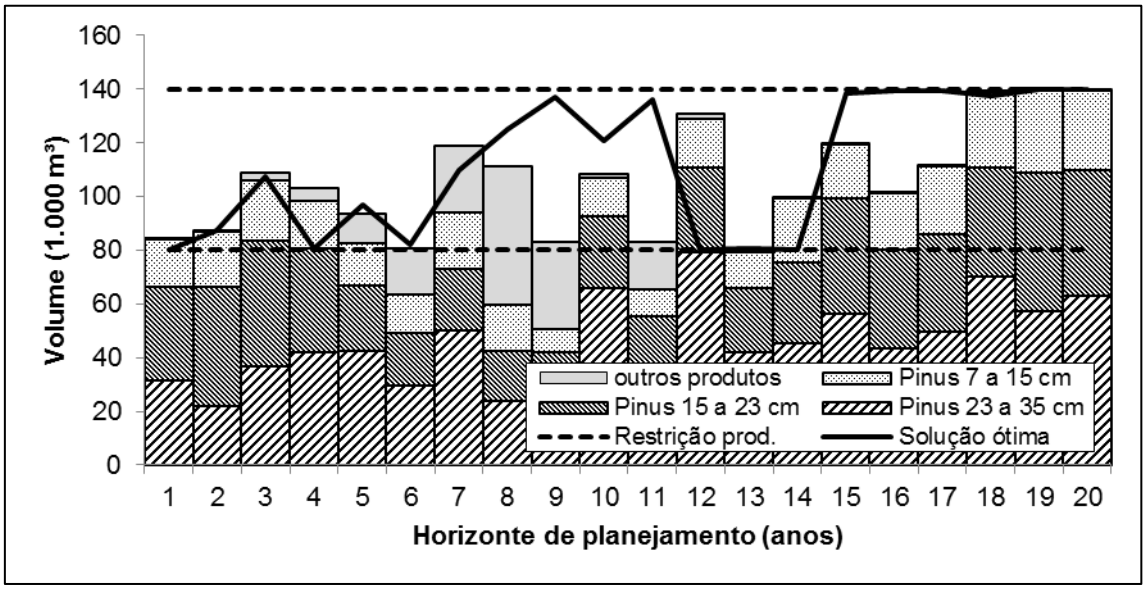


FIGURA 32 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ABORDAGEM IU-SPSO.

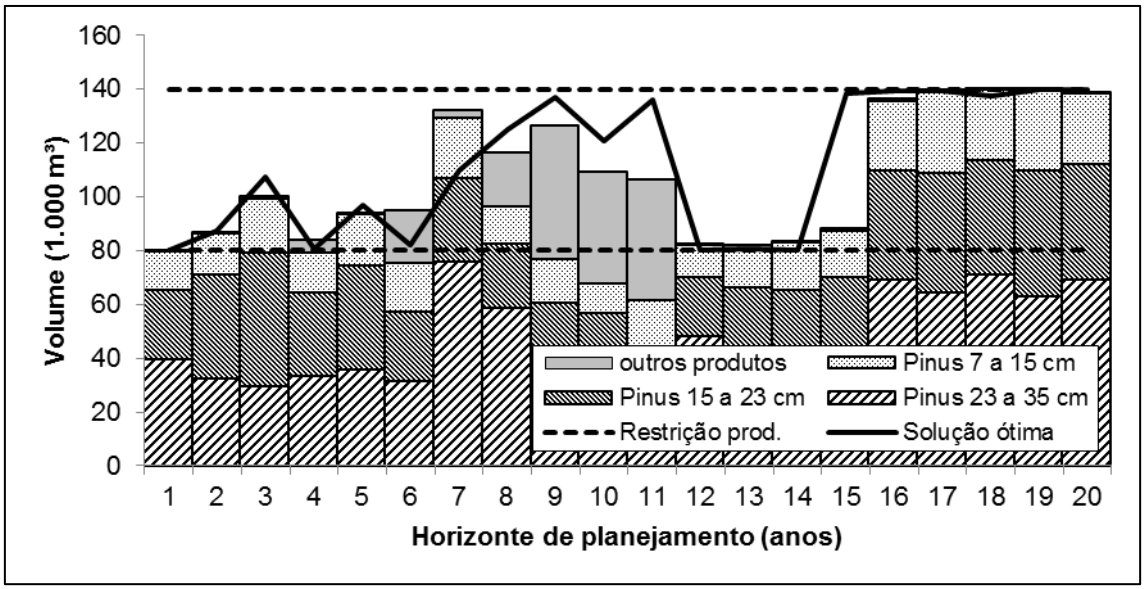


FIGURA 33 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ABORDAGEM N-SPSO.

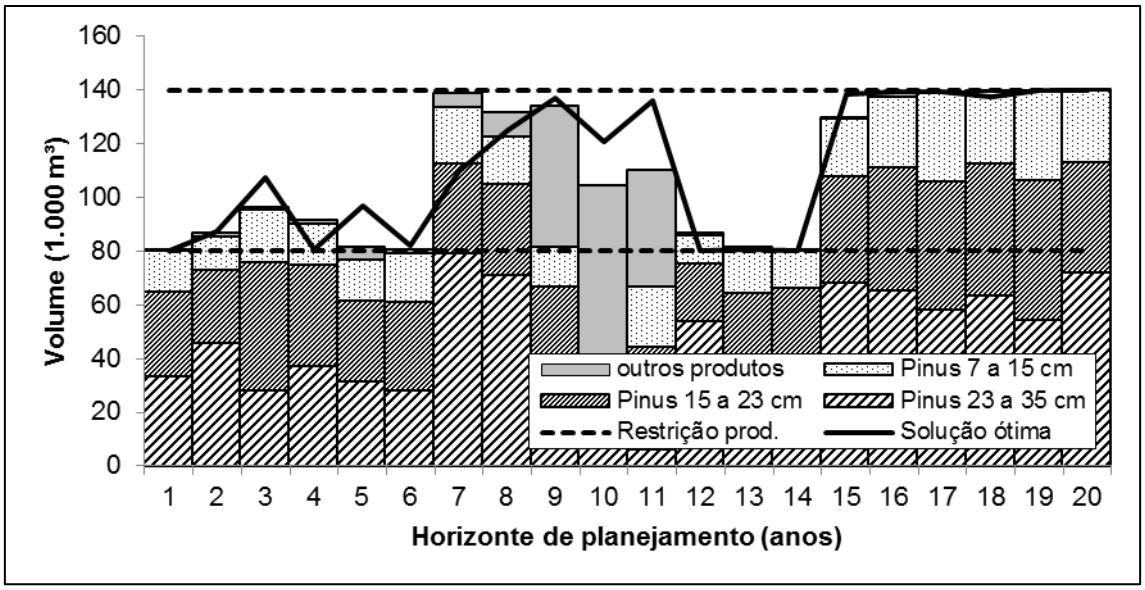


FIGURA 34 - PRODUÇÃO PARA O PROBLEMA ARM_HP20, MELHOR SOLUÇÃO ALGORITMO BT.

As diferenças de resultados para o problema ARM_H20 são estatisticamente significantes. Conforme o teste de Kruskal-Wallis ao nível de significância de 0,05 rejeitou-se a hipótese (p-valor = $4{,}76\times10^{-47}$) de que todas as distribuições de soluções são iguais em todos os grupos testados. Na TABELA 17 são apresentados os coeficientes do resultado do teste estatístico de Dunn. Com base nestes valores é possível afirmar que as mudanças de parâmetros nas meta-heurísticas geram soluções estatisticamente iguais e, ainda, que os algoritmos BT, PSO e a modificação da PSO com vizinhança são diferentes entre si.

TABELA 17 - RESULTADO DO TESTE DE DUNN PARA O PROBLEMA ARM_HP20.

META-HEURÍSTICA	n	$\bar{R}$			
BT(1)	30	225,2	**a***		
BT(2)	30	221,3	**a**		
BT(3)	30	215,0	**a**		
N-SPSO(1)	29	150,2		**b**	
N-SPSO(2)	29	130,4		**b**	
N-SPSO(3)	30	114,4		**b**	
IU-SPSO(1)	30	47,8			**c**
IU-SPSO(2)	29	46,6			**c**
IU-SPSO(3)	28	37,3			**c**

* grupos seguidos de mesma letra não apresentam diferença significativa de acordo com o teste de Dunn ao nível de significância de 0,05

5.8. Avaliação da versão paralela da PSO

Na TABELA 18 são apresentados os valores de tempo médio de processamento em cada problema para diferentes quantidades de processadores. Pode-se perceber o grande potencial de usar a programação paralela principalmente nos tempos apresentados pelo algoritmo SU-PPSO para resolver o problema de maior porte (ARM_HP30). Neste, o tempo de resolução com um processador foi de 2 horas, 26 minutos e 32 segundos, enquanto com oito processadores o tempo médio foi de 28 minutos e 13 segundos.

TABELA 18 - TEMPO MÉDIO DE PROCESSAMENTO DA SU-PPSO.

NUM. PROC.	TEMPO MÉDIO DE PROCESSAMENTO (SEGUNDOS)					
	URM_HP20	URM_HP25	URM_HP30	ARM_HP20	ARM_HP25	ARM_HP30
1	178	563	1340	969	4962	8792
2	98	297	701	483	2493	4546
3	88	222	509	349	1783	3316
4	61	171	383	282	1359	2562
5	61	155	360	253	1231	2290
6	54	137	321	222	1116	2056
7	50	121	288	198	1022	1969
8	46	109	263	177	928	1693

Na FIGURA 35 são apresentados os valores de *speedup* para os problemas avaliados. Como esperado, o algoritmo SU-PPSO síncrono se torna menos eficiente quando mais processadores são adicionados. Resultados semelhantes, com algoritmos PSO paralelos síncronos, também foram encontrados por Venter e Sobieszczanski-Sobieski (2005) e Schutte *et al.* (2004).

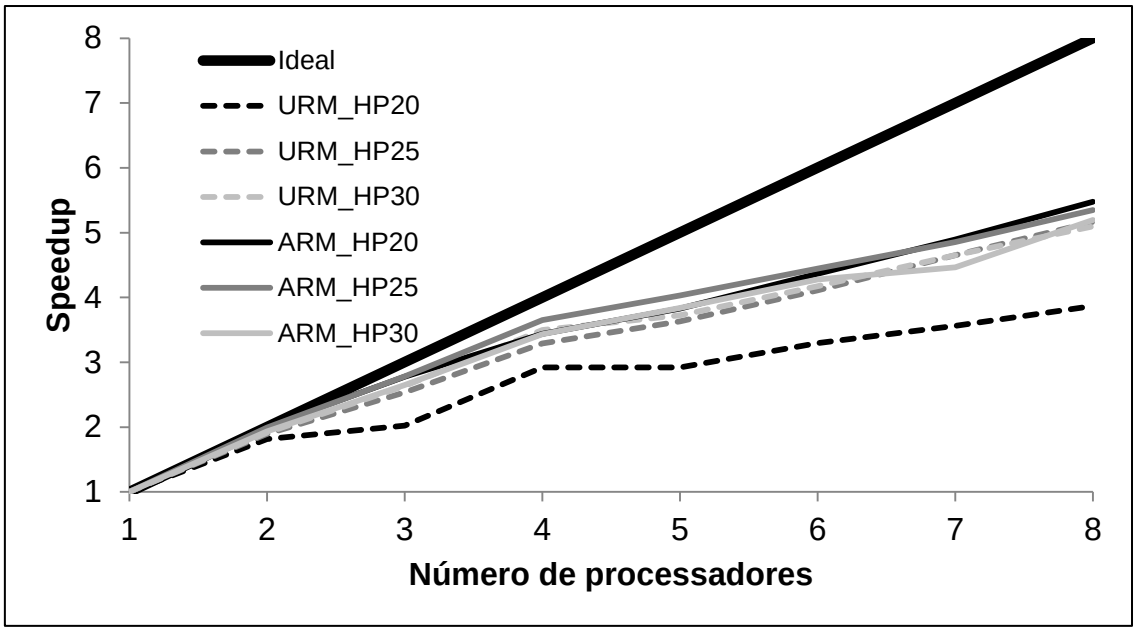


FIGURA 35 - COMPARAÇÃO DA *SPEEDUP* PARALELA DA SU-SPSO.

Na FIGURA 36 é apresentada a eficiência paralela para os problemas avaliados. Em todos os casos verificou-se comportamento decrescente da eficiência paralela do algoritmo SU-PPSO em função do aumento da quantidade de processadores. Ao final de cada iteração o algoritmo é obrigado a sincronizar todos os *threads* que estão sendo utilizados, quanto mais processadores são utilizados, maior é a probabilidade destes ficarem ociosos esperando os processadores que ainda não realizaram todos os seus cálculos.

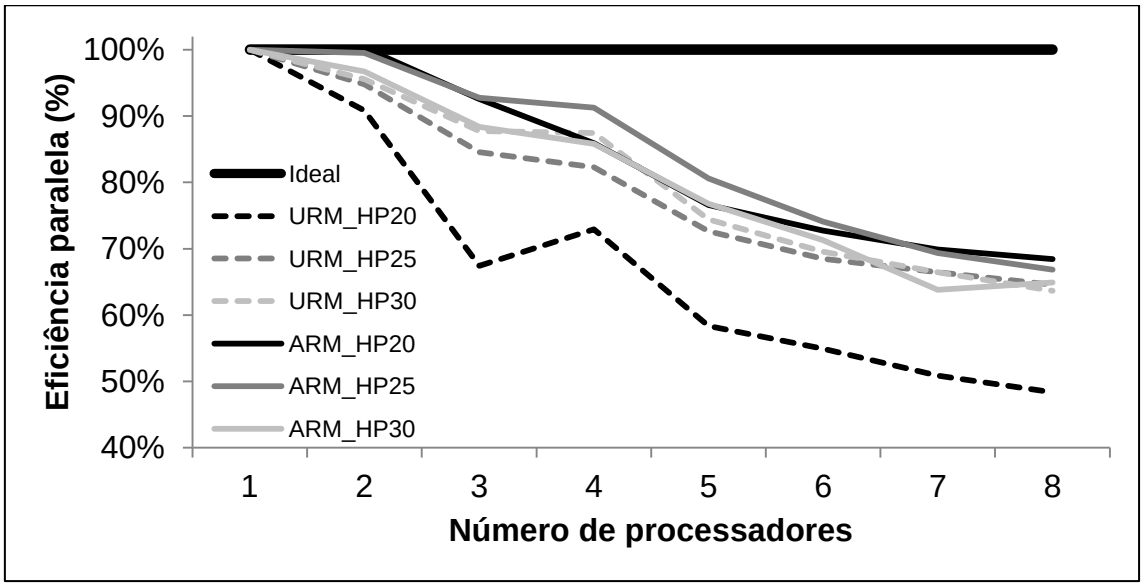


FIGURA 36 - EFICIÊNCIA DE PARALELIZAÇÃO DA SU-PPSO.

Koh *et al.* (2006) avaliaram versões paralelas síncronas e assíncronas do algoritmo PSO em clusters de até 20 computadores. Para a versão síncrona, tal como a utilizada no presente trabalho, os melhores valores de eficiência de paralelização foram obtidos utilizando menos de cinco processadores

5.9. Avaliação da versão paralela da BT

Na TABELA 19 são apresentados os tempos de processamento da versão paralela do algoritmo BT. As versões paralelas dos algoritmos BT e PSO exibiram tempo de processamento semelhante, indicando que o cálculo do *fitness* das soluções é a parte do código de maior custo computacional, independente das estruturas de funcionamento de cada algoritmo.

TABELA 19 - TEMPO MÉDIO DE PROCESSAMENTO DA BT PARALELA.

NUM	TEMPO MÉDIO DE PROCESSAMENTO (SEGUNDOS)					
PROC.	URM_HP20	URM_HP25	URM_HP30	ARM_HP20	ARM_HP25	ARM_HP30
1	169	575	1406	964	4984	9297
2	85	292	708	486	2508	4696
3	57	199	472	323	1767	3263
4	45	156	368	253	1358	2533
5	46	152	368	254	1266	2364
6	38	127	318	214	1082	2004
7	36	121	297	202	1042	2116
8	34	113	260	194	954	1749

Conforme pode ser observado na TABELA 19, para o problema de maior porte, a BT paralela apresentou melhoria no tempo de processamento de 2 horas, 34 minutos e 57 segundos com um processador e 29 minutos e 9 segundos com 8 processadores, o que representa uma diminuição de aproximadamente cinco vezes no tempo de solução.

Na FIGURA 37 é apresentado o comportamento da *speedup* da BT paralela. Nota-se que o *speedup* aumenta quase linearmente em função do

aumento da quantidade de processadores até quatro processadores, se aproximando do *speedup* ótimo. Além disso, pode também se observar que quantidades impares de processadores apresentam maior queda para o valor de *speedup*. È importante ressaltar que o processador utilizado no presente trabalho (modelo i7-3610QM) possui quatro núcleos que simulam oito *threads*, por isso é de se esperar que os melhores desempenhos sejam conseguidos empregando até quatro processadores.

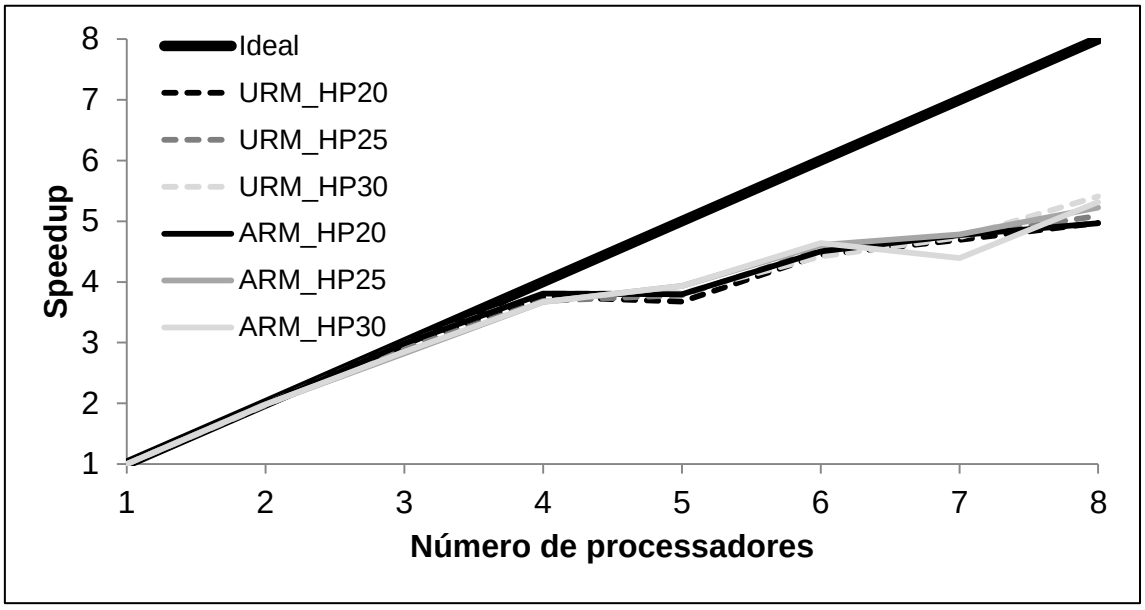


FIGURA 37 - COMPARAÇÃO DA *SPEEDUP* DA BT PARALELA.

Na FIGURA 38 é apresentada a eficiência de paralelização da BT paralela. Pode-se observar que os melhores valores foram obtidos empregando até quatro processadores.

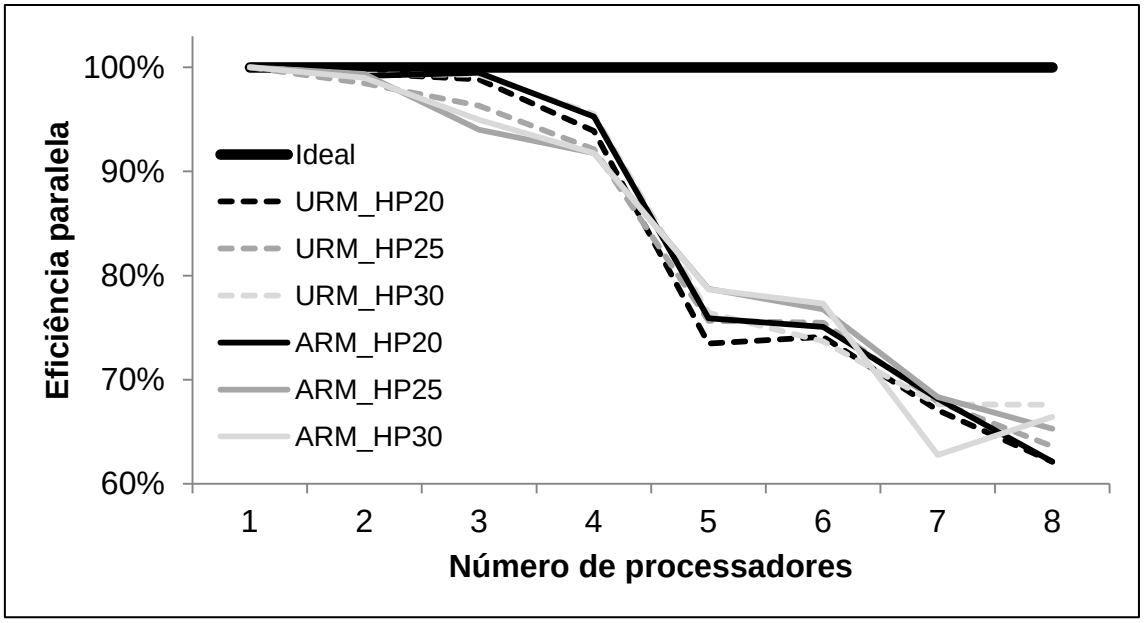


FIGURA 38 - EFICIÊNCIA DE PARALELIZAÇÃO DA BT PARALELA.

6. CONCLUSÕES

- Cenários ARM geram mais restrições e consequentemente o tempo para gerar a solução é maior quando comparado com os cenários URM;
- O aumento da complexidade dos cenários comprova a dificuldade de encontrar solução para os problemas de planejamento florestal de grande porte com os algoritmos exatos. Enquanto as meta-heurísticas fornecem soluções próximas ao ótimo global em tempo viável;
- O algoritmo PSO, tanto na abordagem com atualização imediata quanto com atualização por revoada, apresenta melhores médias quando empregado valores baixos para o V_{max}, mais especificamente, entre a 10% e 20%;
- As abordagens do algoritmo PSO (IU-SPSO e SU-SPSO) são diferentes em relação à qualidade das soluções encontradas;
- O algoritmo BT apresenta melhores resultados com a estratégia de busca na vizinhança total em cada iteração e melhor desempenho médio com vizinhança parcial.
- A modificação proposta neste trabalho ao algoritmo PSO com busca em vizinhança apresenta melhores resultados para o problema de planejamento florestal que o algoritmo PSO original.
- O algoritmo BT, em função de seu potencial de encontrar soluções de alta qualidade, sua simplicidade de implementação e poucos parâmetros de configuração que precisam ser avaliados, é uma opção mais interessante que o algoritmo PSO. Deve-se levar em conta também que o algoritmo BT foi implementado neste trabalho em sua forma mais clássica e que melhorias tais como as estratégias de diversificação e intensificação podem gerar ainda melhores soluções.
- A utilização da programação paralela para solução de problemas de planejamento florestal é uma alternativa promissora, principalmente

para cenários de grande porte com grande quantidade de talhões, opções de manejo e longos horizontes de planejamento.

- A eficiência de paralelização dos algoritmos PSO e BT paralelos, implementados de forma síncrona conforme o modelo *fork-and-join*, é decrescente em função do aumento do número de processadores utilizados. Sugere-se para os próximos trabalhos a implementação dos algoritmos conforme o modelo mestre-escravo como forma de melhorar a eficiência de paralelização.

REFERÊNCIAS

ACKOFF, R. L. e SASIENI, M. W. **Pesquisa operacional**. São Paulo: Livros Técnicos e Científicos Ltda., 1971.

ALONSO, L. R. L. **O problema da consideração de restrições de adjacência em um planejamento florestal**. 126 f. Dissertação (Mestrado em Método Numéricos em Engenharia) - Universidade Federal do Paraná, Curitiba, Paraná 2003.

AMD. Nosso histórico. Disponível em <http://www.amd.com/pt-br/who-we-are/corporate-information/history>. Acesso em 15/05/2014.

ARCE, J. E. **Pesquisa operacional para fins florestais**. Apostila da disciplina de Pesquisa Operacional para Fins Florestais da Pós-Graduação em Ciências Florestais. Curitiba, Paraná, 2007.

BANKS, A.; VINCENT, J. e ANYAKOHA, C. A review of particle swarm optimization. Part I: background and development. **Nat Comput**, v.6, p. 467-484, 2007.

BARRETT, T. M. e GILLESS, J. K. Even-aged restrictions with sub-graph adjacency. **Annals of Operations Research**, v. 95, p. 159 – 175, 2000.

BASKENT, E. Z. e KELES, S. Spatial forest planning: a review. **Ecological Modelling**, v.188, p.145-173, 2005.

BELAL, M e EL-GHAZAWI, T. Parallel models for particle swarm optimizers. **International Journal on Intelligent Cooperative Information Systems**, v. 4, n. 1, 2004.

BETTINGER, P. BOSTON, K. An Analysis of Monte Carlo Integer Programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. **Forest Science**, v. 45, n. 2, p. 292 – 301, 1999.

BETTINGER, P., BOSTON, K. e SESSIONS, J. Intensifying a heuristic forest harvest scheduling search procedure with 2-opt decision choices. **Canadian Journal of Forestry Research**. v. 29, n.11, p.1784-1792, 1999.

BETTINGER, P.; GRAETZ, D.; BOSTON, K.; SESSIONS, J. e CHUNG, W. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. **Silva Fennica**, 36(2), p.561-584, 2002.

BETTINGER, P.; SESSIONS J. e B, BOSTON K. Using Tabu search to schedule timber harvests subject to spatial wildlife goals for big game. **Ecological Modelling**, v. 94, p. 111 – 123, 1997.

BETTINGER, P.; SESSIONS, J.; BOSTON, K. A review of status and use of validation procedures for heuristic used in forest planning. **International Journal of Mathematical and Computational Forestry & Natural-Resource Sciences**, v. 1, n. 1, p. 26-37, 2009.

BNDES. Banco Nacional de Desenvolvimento Econômico e Social (BNDES). Programa para Redução da Emissão de Gases de Efeito Estufa na Agricultura – Programa ABC. Disponível em <http://www.bndes.gov.br/SiteBNDES/bndes/bndes_pt/Institucional/Apoio_Financeiro/Programas_e_Fundos/abc.html >. Acesso em 14/02/2014.

BOSTON, K. e BETTINGER, P. Combining tabu search and genetic algorithm heuristic techniques to solve spatial harvest scheduling problems. **Forest Science**, v. 48, n.1, p. 35 – 46, 2002.

BROOKS, P. **Particle Swarm and priority representation**. Dissertação (Master of Science) – University of Georgia, Athens, Georgia, 2010.

BUONGIORNO, J. e GILLESS, J. K. **Forest management and economics**. New York: MacMillan Publishing Company. 285p. 1987.

CASTRO, E. G. e TSUZUKI, M. S. G. Simulation optimization using swarm intelligence as tool for cooperation strategy design in 3d predator-prey game. In: CHAN, F. T. S. e TIWARI, M. K. **Swarm intelligence - focus on ant and particle swarm optimization**. Vienna, Austria: I-Tech Education and Publishing, 2007. p. 87 – 100.

CHANDRA, R.; DAGUM, L.; KOHR, D.; MAYDAN, D.; MCDONALD, J. e MENON, R. **Parallel programming in OpenMP**. São Franscisco, USA: Morgan Kaufmann , 2001.

CHAPMAN, B.; JOST, G. e VAN DER PAS, R. **Using OpenMP – portable shared memory parallel programming**. England: The MIT Press, 2008.

CLERC, M. e KENNEDY, J. The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space. **IEEE Transactions on Evolutionary Computation**, v.6, n1, p. 58-73, 2002.

CLUTTER, J. C.; FORTSON, J. C.; PLENAAR, L. V.; BRISTER, G. H. e BAILEY, R. L. **Timber management: a quantitative approach**. 3. ed. New York: Jonh Willey, 1983.

COIN-OR. Welcome to the Cbc home page. Disponível em <https://projects.coin-or.org/Cbc>. Acesso em 31/10/2014.

CUI, S. e WEILE, D. S. Application of a Parallel Particle Swarm Optimization Scheme to the Design of Electromagnetic Absorbers. **IEEE Transactions on Antennas and Propagation**, v. 53, n. 11, p. 3616-3624, 2005.

CZAPIŃSKI, M. An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem on CUDA platform. **Journal of Parallel Distributed Computing**, v. 73, p. 1461–1468, 2013.

CZAPIŃSKI, M. e BARNES, S. Tabu Search with two approaches to parallel flowshop evaluation on CUDA platform. **Journal of Parallel Distributed Computing**, v. 71, p. 802–811, 2011.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v.1, p. 3-18, 2011.

DIAS, A. N. **Um modelo para gerenciamento de plantações de eucalipto submetidas a desbaste**. Viçosa, MG. 147 f. Tese (Doutorado em Ciência Florestal) – Universidade Federal de Viçosa, Viçosa, Minas Gerais, 2005.

EBERHART, R. C. and SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Congress on Evolutionary Computation, 2000, **Proceedings**…, 2000, p. 84-88.

EBERHART, R. e SHI, Y. **Computacional intelligence: concepts to implementation**. Burlington, USA: Elsevier, 2007.

ESPOSITO, A. GOMES, H. M. MIGUEL, L. F. F. Computação paralela e serial aplicadas à otimização por enxame de partículas em problemas de engenharia. **Mecánica Computacional**, v XXXI, 2012.

FIECHTER, C. N. A parallel tabu search algorithm for large traveling salesman problems. **Discrete Applied Mathematics**, v. 51, p. 243-267, 1994.

GARCÍA, S.; MOLINA, D.; LOZANO, M.; HERRERA, F.. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. **Journal of Heuristic**, v. 15, p. 617-644, 2009.

GLOVER, F. e LAGUNA, M. **Tabu Search**. Boston: Kluwer, 1997.

GLOVER, F. Future paths for integer programming and links to artificial intellig ence. **Computers e Operations Research**, v.13, p. 533-549, 1986.

GLOVER, F. Tabu search – Part I. **ORSA Journal on Computing,** v.1, n.3, p. 190-206, 1989.

GLPK. Introduction to GLPK. Disponível em: <https://www.gnu.org/software/glp k/#TOCintroduction>. Acesso em 31/10/2014.

GOLDBARG, M. C. e LUNA, H. P. L. **Otimização combinatória e programação linear: modelos e algoritmos**. 2. ed. Rio de Janeiro: Campus, 2005.

GOMIDE, L. R. **Planejamento Florestal Espacial**. 255 f. Tese (Doutorado em Engenharia Florestal) – Universidade Federal do Paraná, Curitiba, Paraná, 2009.

GOMIDE, L. R., ARCE, J. E. e SILVA, A. C. L. Uso do algoritmo genético no planejamento florestal considerando seus operadores de seleção. **Cerne**, Lavras, v. 15, n. 4, p. 460-467, 2009.

GOMIDE, L. R.; ARCE, J. E. e SILVA, A. C. L. Comparação entre a meta-heurística *simulated annealing* e a programação linear inteira no agendamento da colheita florestal com restrições de adjacência. **Ciência Florestal**, Santa Maria, v. 23, n. 2, p. 449-460, 2013.

GUROBI OPTIMIZATION. Gurobi Optimizer 5.6 Overview. Disponível em <http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview>. Acesso em 31/10/2014.

HEPPNER, F. e GRENANDER, U. A stochastic nonlinear model for coordinated bird flocks. In KRASNER, S., Ed., **The Ubiquity of Chaos**. AAAS Publications, Washington, DC, 1990.

IBM. Introdução ao IBM ILOG CPLEX Optimization Studio V12.6.0. Disponível em <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r6/index.jsp?topic=%2Filog.odms.studio.help%2FOptimization_Studio%2Forientation_guide%2Ftopics%2Forientation_intro.html>. Acesso em 31/10/2014.

INTEL. Intel Timeline: A History of Innovation. Disponível em <http://www.intel.com/content/www/us/en/history/historic-timeline.html>. Acesso em 15/05/2014.

JOHNSON, K. N., SCHEURMAN, H. L. Techniques for prescribing optimal timber harvest and investment under different objectives - discussion and synthesis. **Forest Science**, Monograph 18, p.1-31, 1977.

JOHNSTON, D. R.; GRAYSON, A. J. e BRADLEY, R. T. **Planeamento florestal**. Lisboa: Fundação Calouste Gulbenkian, 1977.

KENNEDY, J. e EBERHART, R. C. Particle swarm optimization. In: **IEEE International Conference on Neural Networks**, Perth, Austrália, p. 1942-1948, 1995.

KENNEDY, J. F., EBERHART, R. C. e SHI, Y. **Swarm intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2001.

KIM, J; MUN, K.; KIM, H. e PARK, J. H. Optimal power system operation using parallel processing system and PSO algorithm. **Electrical Power and Energy Systems**, v. 33, p. 1457–1461, 2011.

KOH, B; GEORGE, A. D.; HAFTKA, R. T eFREGLY, B. J. Parallel asynchronous particle swarm optimization. **International Journal for Numerical Methods in Engineering**, v. 67, p. 578–595, 2006.

LACHTERMACHER, G. **Pesquisa operacional na tomada de decisões**. Rio de Janeiro, RJ: Elsevier, 2007.

LI, Y. CAO, Y. LIU, Z. LIU, Y. JIANG, Q. Dynamic optimal reactive power dispatch based on parallel particle swarm optimization algorithm. **Computers and Mathematics with Applications**, v 57, p-1835-1842, 2009.

LINGO. **LINGO user's guide**. Illinois, EUA: LINGO Systems Inc., 2010.

LIU, G.; HAN, S.; ZHAO, X; NELSON, J. D.; WANG, H. e WANG, W. Optimisation algorithms for spatially constrained forest planning. **Ecological Modelling**, v. 194, n. 4, p. 421-428, 2006.

LIU, H; ABRAHAM; A. e GROSAN, C. A Novel Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-shop Scheduling Problems. In.: 2nd International Conference on Digital Information Management, ICDIM '07, p.138-145, 2007.

LOCKWOOD, C. e MOORE, T. Harvest scheduling with spatial constraints: a simulated annealing approach. **Canadian Journal of Forestry Research**, v. 23, n. 3, p. 468-478, 1993.

MALEK, M.; GURUSWAMY, M. e PANDYA, M. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. **Annals of Operations Research**, v. 21, p. 59-84, 1989.

MCDILL, M. E. e BRAZE, J. Comparing adjacency constraint formulations for randomly generated forest planning problems with four age class distribution. **Forest Science**, v.46, n.3, 2000.

MCDILL, M. E.; REBAIN, S. A. e BRAZE, J. Harvest Scheduling with area-based adjacency constraint. **Forest Science**, USA, v.48, n.4, p.631-642, 2002.

MEDEIROS, J. A. C. C. **Enxame de partículas como ferramenta de otimização em problemas complexos de engenharia nuclear**. 119 f. Tese (Doutorado em Ciências em Engenharia Nuclear) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 2005.

MEGGINSON, L. C.; MOSLEY, D. C. e PIETRI Jr, P. H. **Administração – conceitos e aplicações**. 4. ed.. São Paulo: Harbra, 1998.

MENDES, R. **Population topologies and their influence in particle swarm performance**. 2004. 189 f. Dissertação. Universidade do Minho, Portugal, 2004.

MORAES, A. O. S. **Desenvolvimento e implementação de versões paralelas do algoritmo do enxame de partículas em clusters utilizando MPI**. 154 f. Dissertação (Mestrado em Engenharia Química) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 2011.

MPI FORUM. The Message Passing Interface (MPI) standard. Disponível em <http://www.mcs.anl.gov/research/projects/mpi/>. Acesso em 15/05/2014.

MÜLLER, V. **Otimização de layouts industriais através do método enxame de partículas**. 79 f. Dissertação (Mestrado em Controle e Otimização de Processos Industriais) – Universidade de Santa Cruz do Sul, Santa Cruz do Sul, Rio Grande do Sul, 2007.

MURRAY, A. T. Spatial restrictions in harvest scheduling. **Forest Science**, v.45, n.1, 1999.

NAKA, S.; FUKUYAMA, Y.; GENJI, T. e YURA, T. Practical Distribution State Estimation Using Hybrid Particle Swarm Optimization. In: IEEE Power Engineering, 2001, **Proceedings** … Columbus, Ohio, USA, 2001, p. 1- 6.

NÁPOLES, G; GRAU, I. e BELLO, R. Constricted Particle Swarm Optimization based Algorithm for Global Optimization. **Polibits**, v.46, p.5-11, 2012.

NASCIMENTO, F. A. F.; ARCE, J. E.; DIAS, A. N.; FIGUEIREDO FILHO, A.; MIRANDA, G. M. e CARNIERI, C. Meta-heurística otimização por enxame de partículas aplicada ao planejamento florestal. In.: LOPES, H. S.; RODRIGUES, L. C. A. e STEINER, M. T. A. (Eds.) **Meta-heurística em pesquisa operacional**. Curitiba, Paraná: Ominipax, 2013, p. 355 – 366.

NASCIMENTO, F. A. F.; DIAS, A. N.; FIGUEIREDO FILHO, A.; ARCE, J. E. e MIRANDA, G. M. Uso da Meta-heurística otimização por enxame de partículas no planejamento florestal. **Scientia Forestalis**, v. 40, n. 96, p. 557-565, 2012.

NASCIMENTO, F. A. F.; GUERRA, F. A. e COELHO, L. S. Abordagem de enxame de partículas aplicada à engenharia de confiabilidade. In: Induscon – VII Conferência Internacional de Aplicações Industriais, 2008, **Anais**... Poços de Caldas, 2008.

NASCIMENTO, L. S. V.; REIS JÚNIOR, D. S. e MARTINS, E. S. P. R. Comparação de algoritmos evolucionários na otimização multiobjetivo de sistemas de reservatórios. In.: XVII Simpósio Brasileiro de Recursos Hídricos, 2007, **Anais**... São Paulo, 2007.

NVIDIA. CUDA FAQ. Disponível em <https://developer.nvidia.com/cuda-faq>. Acesso em 02/06/2014.

O'HARA, A. J.; FAALAND, B. H e BARE, B. B. Spatially constrained timber harvest scheduling. **Canadian Journal of Forest Research**, v. 19, p.715 – 724, 1989.

OLIVEIRA, E. B. ***Softwares*** **para manejo e análise econômica de plantações florestais**. Documentos 216, Embrapa Florestas, Colombo, 68p., 2011.

OPENMP ARB – OpenMP Architeture Review Board. The OpenMP® API specification for parallel programming. Disponível em <http://openmp.org/wp/about-openmp/>. Acesso em 05/10/13.

OPTIMBER. OpTimber: Otimizador de manejo florestal de longo prazo, versão 2.3.181. Curitiba, Optimber Otimização e Informática Ltda. 2013.

OPTIMBER. **Produtos: OpTimber-LP**. Disponível em <http://www.optimber.com.br/index.php/softwares/optimber-lp>. Acesso em 14/02/2014.

PARANÁ (Estado). Secretaria de Estado da Agricultura e do Abastecimento. Departamento de Economia Rural. **Preços Florestais**. Disponível em <http://www.agricultura.pr.gov.br/modules/conteudo/conteudo.php?conteudo=129>. Acesso em 14/02/2014.

PUKKALA, T. e HEINONEN, T. A Comparison of One- and Two- Compartment Neighborhoods in Heuristic Search with Spatial Forest Management Goals. **Silva Fennica**, v. 38, n.3, p. 319-332, 2004.

R Core Team (2013). **R: A language and environment for statistical computing**. R Foundation for Statistical Computing, Vienna, Áustria. Disponível em <http://www.R-project.org/>. Acesso em 13/02/2014.

REEVES, W. T. Particle Systems-A Technique for Modeling a Class of Fuzzy Objects. **Acm Transactions on Graphics**, v2, n2, 1983. E republicada em Computer Graphics. v17 n3, 1983, (**acm SIGGRAPH `83 Proceedings**), pp. 359-376.

REYNOLDS, C. W. Flocks, herds and schools: a distributed behavioral model. **Computer Graphics**, 21(4):25-34, 1987.

REZENDE, J. L. P. e OLIVEIRA, A. D. **Análise econômica e social de projetos florestais**. Viçosa, MG: Imprensa Universitária, UFV, 2001.

RICHARDS, E. W. e GUNN, E. A model and tabu search method to optimize stand harvest and road construction schedules. **Forest Science**, v. 46, n. 2, p. 188-203, 2000.

RODRIGUES, F. L.; LEITE H. G.; SANTOS, H. N. e SOUZA, A. L. Soluções de problemas de planejamento florestal com restrições de inteireza utilizando Busca Tabu. **Revista Árvore**, Viçosa, v.27, n.5, p.701-713, 2003.

RODRIGUES, F. L.; LEITE H. G.; SOUZA, A. L.; RIBEIRO, C. A. A. S. e SILVA, M. L. Regulação de florestas equiâneas utilizando programação linear: uma

aplicação da teoria do modelo II. **Revista Árvore**, Viçosa, v.22, n.5, p.193-213, 1998.

SABAT, S. L.; ALI, L. e UDGATA, S. K. Integrated Learning Particle Swarm Optimizer for global optimization. **Applied Soft Computing**, v.11, p. 574–584 , 2011.

SANDRINI, F. T. **Otimização de Banco de Capacitores em Sistemas de Distribuição de Energia Elétrica Usando Algoritmos Genéticos e Nuvem de Partículas**. 112 f. Dissertação (Mestrado em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná, Curitiba, Paraná, 2005.

SCHNEIDER, P. R. e DURLO, M. A. Avaliação florestal. **Centro de Pesquisas Florestais, Série Técnica nº 02**, Santa Maria, 1987.

SCHUTTE , J. F.; REINBOLT , J. A.; FREGLY , B. J.; HAFTKA, R. T. E GEORGE A. D. Parallel global optimization with the particle swarm algorithm. **Intenational Journal Numerical Methods Engineering**, v. 61, p. 2296–2315, 2004.

SCHUCHOVSKI, M. S. **Diagnóstico e planejamento do consumo de madeira e da produção em plantações florestais no estado do Paraná**. 90 f. Dissertação (Mestrado em Ciências Florestais) Programa de Pós-Graduação em Engenharia Florestal, Universidade Federal do Paraná, Curitiba, Paraná, 2003.

SHA, D. Y e HSU, C. A new particle swarm optimization for the open shop scheduling problem. **Computers e Operations Research**, v. 35, p. 3243 – 3261, 2008.

SHAN, Y; BETTINGER, P.; CIESZEWSKI, C. e WANG, W. Pitfalls and potential of particle swarm optimization for contemporary spatial forest planning. **Forest Systems**, v.21, n.3, p468-480, 2012.

SHI, Y. e EBERHART. R. A modified particle swarm optimizer. **Proceedings of the IEEE international conference on evolutionary computation**, p. 69–73. IEEE Press, Piscataway, NJ, 1998.

SHIMIZU, Y. e KAWAMOTO, H. An Implementation of Parallel Computing for Hierarchical Logistic Network Design Optimization Using PSO. In: 18th European Symposium on Computer Aided Process Engineering – ESCAPE 18, 2008, **Proceeding**…Elsevier, 2008, p. 605 - 611.

SIERAKOWSKI, C. A. e COELHO, L. S. A software tool for teaching of particle swarm optimization fundamentals. **Advances in Engineering Software**, v. 39, n.11, p. 877-887, 2008.

SIERAKOWSKI, C. A. **Inteligência coletiva aplicada a problemas de robótica móvel**. 160 f. Dissertação (Mestrado em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná, Curitiba, Paraná, 2006.

SILVA, G. F.; GHISOLFI, E. M.; TEIXEIRA, A. F.; CABRINI, A. M. e BARROS JÚNIOR, A. A. O método das restrições na solução de um problema de planejamento florestal multiobjetivo**. Revista Brasileira de Ciências Agrárias** v.1, p.41-48, 2006.

SILVA, G. F.; LEITE, H. G.; SILVA, M. L. RODRIGUES, F. L. e SANTOS, H. N. Problemas com uso de programação linear com posterior arredondamento da solução ótima, regulação florestal. **Revista Árvore**, Viçosa-MG, v.27, n.5, p.677-688, 2003.

SILVA, M. L.; JACOVINE, L. A. G. e VALVERDE, S. R. **Economia florestal**. Viçosa: Imprensa Universitária, UFV, 2005.

SILVA, R. T. **Planeamento florestal, modelos de programação inteira multiobjectivo e aplicações**. 132 f. Dissertação (Mestrado em Gestão da Informação nas Organizações) - Universidade de Coimbra, Coimbra, Portugal, 2004.

SMYTHE, J. **Roulette wheel particle swarm optimization**. 51 f. Dissertação (Master of Science) – University of Georgia, Athens, Georgia, 2012.

SZYMON, J. e DOMINIK, Z. Solving Multi-criteria Vehicle Routing Problem by Parallel Tabu Search on GPU. **Procedia Computer Science**, v. 18, p. 2529 – 2532, 2013.

TALBI, E. G.; ZAFIDI, Z e GEIB, J-M. A parallel adaptive tabu search approach. **Parallel Computing**, v. 24, p. 2003-2019, 1998.

TEBALDI, A.; COELHO, L. S. e LOPES JUNIOR, V. Detecção de falhas em estruturas inteligentes usando otimização por nuvem de partículas: fundamentos e estudo de casos. **Revista Controle e Automação**, v.17 n.3, p. 312-330, 2006.

VAN DEUSEN, P. C. Multiple solution harvest scheduling. **Silva Fennica**, v. 33, n. 3, p. 207-216, 1999.

VENDER, G. e SOBIESKI-SOBIESZCZANSKI, J. A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. In: 6th World Congresses of Structural and Multidisciplinary Optimization, 2005, **Anais**… Rio de Janeiro, 2005, p. 2 – 10.

VOSS, S. Meta-heuristic: The state of art. In: NAREYEK, A. (Editor). Local search for planning and scheduling. Berlin: Springer-Verlag, 2001, p.1 - 23.

VOSS, S.; MARTELLO, S.; OSMAN, I. H. e ROUCAIROL, C.. **Meta-Heuristics: advances and trends in local search paradigms for optimization**. Kluwer, Boston, 1999.

WANG, D. WU, C. H. IP, A. WANG, D. YAN, Y. Parallel multi-population particle swarm optimization algorithm for the uncapacitated facility location problem using OpenMP. In: IEEE Congress on Evolutionary Computation, 2008, **Proceeding**… 2008, p. 1214 – 1219.

YOSHIDA, H.; FUKUYAMA, Y.; KAWATA, K.; TAKAYAMA, S. e NAKANISHI, Y. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. **IEEE Transaction on Power Systems**, v.15, n. 4, p. 1232-1239, 2001.

ZAR, J. H. **Biostatistical analysis**. 5. ed. New Jersey, EUA: Pearson Prentice Hall, 2010.

APÊNDICE I – Pseudocódigo para geração de alternativas de manejo.

Pseudocódigo 1. Geração de alternativas de manejo.

```
1   var
2          i, idade, soma_hp, hp, quant_desb, quant_corte : inteiro
3          vec_desb[], vec_corte[], vec_am[] : vetor de inteiro
4   inicio
5          i <- 0
6          se idade < ( vec_desb[quant_desb - 1] ) # se iniciar com desbaste
7                 enquanto i < quant_desb
8                        soma_hp <- soma_hp – idade
9                        vec_am.push_back( vec_desb[i] )
10                       soma_hp <- soma_hp + vec_desb[i]
11                       insere_opcao_manejo(1)
12                       i <- i + 1
13                fimenquanto
14         senao                                    # se iniciar com corte raso
15                enquanto i < quant_corte
16                       soma_hp <- soma_hp – idade
17                       vec_am.push_back( vec_corte[i] )
18                       soma_hp <- soma_hp + vec_corte[i]
19                       insere_opcao_manejo(0)
20                       i <- i + 1
21                fimenquanto
22         fimse
23  fim
24  # função recursiva que insere uma nova opção de manejo no vetor vec_am[]
25  funcao insere_opcao_manejo( dc ) # dc = 0 desbaste, dc = 1 corte raso
26         var
27                i : inteiro
28         se soma_hp > hp
29                se dc = 0           # corte raso na recursão anterior
30                       soma_hp <- soma_hp – vec_am.back()
31                       vec_am.pop_back()
32                       soma_hp <- soma_hp + vec_am.back()
33                senaose dc = 1      # desbaste na recursão anterior
34                       soma_hp <- soma_hp – vec_am.back()
35                       vec_am.pop_back()
36                fimse
37                registra_am()
38                retorna
39         fimse
40
41         se dc = 0
42                enquanto i < quant_desb
43                       vec_am.push_back( vec_desb[i] )
44                       soma_hp <- soma_hp + vec_desb[i]
45
46                       insere_opcao_manejo(1)
47                       i <- i + 1
48                fimenquanto
49         senaose dc = 1
50                enquanto i < quant_corte
51                       vec_am.push_back( vec_corte[i] )
52                       soma_hp <- soma_hp + vec_corte[i]
53
54                       insere_opcao_manejo(0)
55                       i <- i + 1
56                fimenquanto
57         fimse
58  fim
```

O algoritmo do Pseudocódigo 1 utiliza uma função recursiva que adiciona opções de manejo ao final de um vetor, alternando entre desbaste e corte raso até que a soma destas ultrapasse a quantidade de anos do horizonte de planejamento. Para o seu funcionamento é necessário entrar com as informações de idade do talhão (`idade`), tamanho do horizonte de planejamento (`hp`) e as quantidades de opções de desbaste (`quant_desb`) e opções de corte raso (`quant_corte`). Devem ser informadas também todas as possíveis opções de desbaste e corte raso, neste caso, as variáveis são declaradas como vetores (`vec_corte[]` e `vec_desb[]`). As opções de manejo são armazenadas em um vetor (`vec_am[]`) a qual compõe, ao final de cada *loop*, a alternativa de manejo gerada.

Para a entrada da informação idade do talhão considerou-se o valor da quantidade de anos em números inteiros, ou seja, os meses não entram na conta para determinar a idade do talhão.

Na linha 24 a função recursiva `insere_opcao_manejo(dc)` é declarada. A primeira parte desta função verifica se o horizonte de tempo da alternativa de manejo gerada ultrapassa o horizonte de planejamento. Se sim, retira-se do vetor `vec_am[]` a última opção adicionada e diminui-se da variável `soma_hp` o tempo da opção manejo. Depois disso, na linha 45 a alternativa de manejo gerada é registrada por meio da função `registra_am()`. Esta função deve verificar se a AM gerada não é igual a alguma anteriormente armazenada para evitar duplicatas. Depois de registrada a AM, a função `insere_opcao_manejo(dc)` retorna (linha 38), fazendo com que a função volte um nível da recursividade.

Para simplificar a escrita do Pseudocódigo 1 utilizaram-se as seguintes funções de manipulação de vetores: `push_back()`, adiciona um novo elemento ao final do vetor; `pop_back()`, remove o último elemento do vetor; e `back()`, retorna uma referência para o ultimo elemento do vetor.

APÊNDICE II – Exemplo de funcionamento do algoritmo de geração de alternativas de manejo.

O exemplo descrito a seguir considera um manejo no qual o desbaste pode ser realizado entre 9 e 11 anos de idade e o corte raso entre 16 e 19 anos de idade. O talhão considerado possui idade atual de 6 anos e o horizonte de planejamento possui 30 períodos, sendo cada ano um período. O algoritmo inicia verificando se deve iniciar por desbaste ou corte raso. Como a idade é menor que o último desbaste possível, inicia-se pelo desbaste. Adiciona-se na AM a primeira opção possível de desbaste (9) e armazena-se o tempo desta intervenção (3). Logo em seguida é adicionado o primeiro corte raso (16). O tempo agora passa a ser de 10 anos, 3 anos do desbaste mais 7 anos entre esta intervenção e o corte raso. O algoritmo continua adicionando alternadamente desbastes e cortes rasos até que o tempo total da AM seja maior que o HP de 30 anos. Assim, gera-se a seguinte AM: D9 – C16 – D9 – C16 – D9. Como esta AM possui tempo total de 35 anos, retira-se a última opção de desbaste e registra-se a primeira AM:

D9 – C16 – D9 – C16

A próxima opção de desbaste adicionada é de 10 anos. Com isso, tem-se um tempo total de 36 anos. Novamente a última opção é retirada e gera-se a AM D9 – C16 – D9 – C16. Percebe-se que esta AM é uma duplicada da última AM registrada, por isso ela não é registrada. Isso acaba ocorrendo com todas as outras opções de desbaste que serão adicionadas na sequência. Com isso, o algoritmo volta um nível da função recursiva e adiciona a próxima opção de corte raso, 17 anos. Isso acaba gerando a AM D9 – C16 – D10 – C17 – D9. Esta AM também possui tempo total de 35 anos. Novamente retira-se a última opção de desbaste e registra-se a nova AM:

D9 – C16 – D9 – C17

Na sequência o algoritmo gera as demais alternativas com inicio D9 – C16 – D9:

D9 – C16 – D9 – C18

D9 – C16 – D9 – C19

Depois disso, muda-se a opção de desbaste do terceiro nível para D10 e gera-se todas as alternativas com inicio D9 – C16 – D10:

D9 – C16 – D10 – C16

D9 – C16 – D10 – C17

D9 – C16 – D10 – C18

D9 – C16 – D10 – C19

O mesmo procedimento é realizado para o desbaste aos 11 anos no terceiro nível gerando as seguintes alternativas:

D9 – C16 – D11 – C16

D9 – C16 – D11 – C17

D9 – C16 – D11 – C18

D9 – C16 – D11 – C19

Ao final, o algoritmo gera, para as configurações consideradas, as alternativas de manejo apresentadas no quadro a seguir.

D9 - C16 - D9 - C16	D9 - C18 - D11 - C16	D10 - C17 - D11 - C17	D11 - C17 - D9 - C16
D9 - C16 - D9 - C17	D9 - C18 - D11 - C17	D10 - C17 - D11 - C18	D11 - C17 - D9 - C17
D9 - C16 - D9 - C18	D9 - C18 - D11	D10 - C17 - D11	D11 - C17 - D9 - C18
D9 - C16 - D9 - C19	D9 - C19 - D9 - C16	D10 - C18 - D9 - C16	D11 - C17 - D9
D9 - C16 - D10 - C16	D9 - C19 - D9	D10 - C18 - D9 - C17	D11 - C17 - D10 - C16
D9 - C16 - D10 - C17	D9 - C19 - D10 - C16	D10 - C18 - D9	D11 - C17 - D10 - C17
D9 - C16 - D10 - C18	D9 - C19 - D10	D10 - C18 - D10 - C16	D11 - C17 - D10 - C18
D9 - C16 - D10 - C19	D9 - C19 - D11 - C16	D10 - C18 - D10 - C17	D11 - C17 - D10
D9 - C16 - D11 - C16	D9 - C19 - D11	D10 - C18 - D10	D11 - C17 - D11 - C16
D9 - C16 - D11 - C17	D10 - C16 - D9 - C16	D10 - C18 - D11 - C16	D11 - C17 - D11 - C17
D9 - C16 - D11 - C18	D10 - C16 - D9 - C17	D10 - C18 - D11 - C17	D11 - C17 - D11 - C18
D9 - C16 - D11 - C19	D10 - C16 - D9 - C18	D10 - C18 - D11	D11 - C17 - D11
D9 - C17 - D9 - C16	D10 - C16 - D9 - C19	D10 - C19 - D9 - C16	D11 - C18 - D9 - C16
D9 - C17 - D9 - C17	D10 - C16 - D10 - C16	D10 - C19 - D9	D11 - C18 - D9 - C17
D9 - C17 - D9 - C18	D10 - C16 - D10 - C17	D10 - C19 - D10 - C16	D11 - C18 - D9
D9 - C17 - D9	D10 - C16 - D10 - C18	D10 - C19 - D10	D11 - C18 - D10 - C16
D9 - C17 - D10 - C16	D10 - C16 - D10 - C19	D10 - C19 - D11 - C16	D11 - C18 - D10 - C17
D9 - C17 - D10 - C17	D10 - C16 - D11 - C16	D10 - C19 - D11	D11 - C18 - D10
D9 - C17 - D10 - C18	D10 - C16 - D11 - C17	D11 - C16 - D9 - C16	D11 - C18 - D11 - C16
D9 - C17 - D10	D10 - C16 - D11 - C18	D11 - C16 - D9 - C17	D11 - C18 - D11 - C17
D9 - C17 - D11 - C16	D10 - C16 - D11 - C19	D11 - C16 - D9 - C18	D11 - C18 - D11
D9 - C17 - D11 - C17	D10 - C17 - D9 - C16	D11 - C16 - D9 - C19	D11 - C19 - D9 - C16
D9 - C17 - D11 - C18	D10 - C17 - D9 - C17	D11 - C16 - D10 - C16	D11 - C19 - D9
D9 - C17 - D11	D10 - C17 - D9 - C18	D11 - C16 - D10 - C17	D11 - C19 - D10 - C16
D9 - C18 - D9 - C16	D10 - C17 - D9	D11 - C16 - D10 - C18	D11 - C19 - D10
D9 - C18 - D9 - C17	D10 - C17 - D10 - C16	D11 - C16 - D10 - C19	D11 - C19 - D11 - C16
D9 - C18 - D9	D10 - C17 - D10 - C17	D11 - C16 - D11 - C16	D11 - C19 - D11
D9 - C18 - D10 - C16	D10 - C17 - D10 - C18	D11 - C16 - D11 - C17	
D9 - C18 - D10 - C17	D10 - C17 - D10	D11 - C16 - D11 - C18	
D9 - C18 - D10	D10 - C17 - D11 - C16	D11 - C16 - D11 - C19	

APÊNDICE III – Tabela de produção para os povoamentos *Pinus sp* em crescimento no final do horizonte de planejamento.

Desb.	Idade final	Vol (m³/ha) na idade final		
		7 a 15* (cm)	15 a 23 (cm)	23 a 35 (cm)
-	9	133,10	50,10	0,00
-	10	141,50	82,00	0,00
-	11	132,80	137,80	0,00
8	9	71,60	46,10	0,00
8	10	75,70	75,90	0,00
8	11	62,30	118,90	0,50
8	12	67,70	132,50	14,70
8	13	62,00	153,10	27,70
8	14	55,20	174,90	43,50
8	15	57,60	184,50	60,90
8	16	59,10	190,40	79,70
8	17	65,30	194,50	98,80
8	18	49,90	213,30	119,60
8	19	51,80	214,10	139,70
8	20	53,70	218,80	159,40
9	10	63,70	85,50	0,00
9	11	63,40	115,20	0,30
9	12	68,90	131,10	11,90
9	13	55,00	163,70	23,90
9	14	56,40	175,30	38,90
9	15	59,20	185,20	55,70
9	16	64,20	191,40	74,00
9	17	57,80	205,20	92,80
9	18	51,10	215,50	112,80
9	19	57,40	216,60	133,00
9	20	55,60	221,20	152,70
10	11	64,30	110,80	0,10
10	12	70,00	129,10	9,10
10	13	57,40	162,90	20,10
10	14	57,70	175,30	34,10
10	15	60,60	185,50	50,30
10	16	66,00	192,10	67,90
10	17	58,70	206,90	86,40
10	18	54,00	217,50	105,90
10	19	58,90	219,00	125,70
10	20	57,30	223,40	145,10
11	12	70,90	126,50	6,50
11	13	58,90	161,40	16,20
11	14	61,40	175,00	29,10
11	15	61,90	185,60	44,60
11	16	67,50	192,60	61,50
11	17	54,20	216,10	79,60
11	18	55,30	219,30	98,50
11	19	60,10	221,10	117,80
11	20	62,00	225,30	136,90
12	13	60,50	159,30	12,50
12	14	62,40	174,20	24,40
12	15	67,10	185,40	39,00
12	16	57,00	204,80	55,20
12	17	55,50	216,90	72,60
12	18	59,20	220,70	90,90
12	19	58,60	225,20	109,30
12	20	63,50	226,90	128,20

* diâmetro da tora (cm)

APÊNDICE IV – Pseudocódigo para o algoritmo *path* (basedo em McDill *et al.*, 2002).

Pseudocódigo 2. Algoritmo path (basedo em McDill *et al.*, 2002)

```
1   var
2         a, areaMax, tamCluster, tamPath, numNos : inteiro
3         cluster[], path[] : vetor de inteiro
4         No[] : vetor da classe CNo
5   Inicio
6         tamCluster <- tamPath <- 0
7         clusterInicial()      # gera o cluster inicial
8         a <- proximoNo()      # escolhe o próximo nó a entrar no cluster
9         enquanto a <> -1
10              adicionaAoPath( a )        # adiciona o nó "a" no path[]
11              adicionaNo(a, 0)           # função recursiva
12              adicionaAoCluster( a )     # aciona o nó "a" ao cluster[]
13              a <- proximoNo()           # próximo nó a entrar no cluster[]
14        fimenquanto
15  fim
16  funcao adicionaNo(a, area)
17        var b, i : inteiro
18        area <- area + No[a].area
19        se area > areaMax
20              registraPath()
21              retorna
22        fimse
23        para i de 1 ate tamCluster faca
24              b <- cluster[i]
25              se No[a].vizinho(b) = verdadeiro e estaNoPath(b) = falso
26                    adicionaAoPath( b )
27                    adicionaNo(b, area)
28              fimse
29        fimpara
30  fim
31  funcao proximoNo()
32        para i de 1 ate numNos faca
33              se estaNoCluster(i) = falso
34                    para j de 1 ate tamCluster faca
35                          se No[cluster[j]].vizinho(i) = verdadeiro
36                                retorna i
37                          fimse
38                    fimpara
39              fimse
40        fimpara
41        retorna -1
42  fim
43  classe No #Classe nó
44        var
45              area : real
46              vizinhos[] : vetor de inteiro
47              numVizinhos : inteiro
48        funcao vizinho(no)
49              para i de 1 ate numVizinhos faca
50                    se vizinhos[i] = no
51                          retorne verdadeiro
52                    fimse
53              fimpara
54              retorne falso
55        fim
56  fim
```

O algoritmo apresentado no Pseudocódigo 2 tem a seguintes funções auxiliares:

- `clusterInicial()`: (linha 7) gera o *cluster* inicial conforme o passo (1) do algoritmo proposto por McDill *et al.* (2002);
- `adicionaAoPath(a)`: (linha 10) adiciona o nó “a” ao vetor `path[]` e incrementa o valor da variável `tamPath`;
- `adicionaAoCluster(a)`: (linha 12) adiciona o nó “a” ao vetor `cluster[]` e incrementa o valor da variável `tamCluster`;
- `estaNoPath(b)`: (linha 25) retorna verdadeiro se o nó “b” está no *cluster* e falso caso contrário.
- `registraPath()`: (linha 20) verifica se o path gerado não é redundante a algum *path* anteriormente gerado, se não, registra o novo path em uma lista específica.

APÊNDICE V – Modelo LINGO

```
MODEL:
TITLE: Problema de planejamento florestal com restrições de adjacência;

SETS:
       UG;                          !Unidades de gestão (talhões);
       AM;                          !alternativas de manejo;
       HP /HP1..HP20/;              !horizonte de planejamento (20 anos);

       UG_AM(UG, AM): VPE, CORTE;
       UG_AM_HP(UG, AM, HP): VOL, DC;
       V_HP(HP): VOL_HP;
       AIJ(UG,UG): ADJ;
ENDSETS

DATA:
       UG  = @FILE('LingoDados_ID_VPE.txt');
       AM  = @FILE('LingoDados_ID_VPE.txt');
       VPE = @FILE('LingoDados_ID_VPE.txt');
       VOL = @FILE('LingoDados_VOL.txt');
       DC  = @FILE('LingoDados_DESBCORTE.txt');
       ADJ = @FILE('LingoDados_Matriz_aij.txt');

       VOL_MIN = 40000;
       VOL_MAX = 180000;
ENDDATA

!Função Objetivo;
[FO] MAX = @SUM(UG(I): @SUM(AM(J): VPE(I,J) * CORTE(I,J)));

!Restrição de adjacência;
@FOR(HP(K): ! para cada ano do HP;
       @FOR(UG(I): ! para cada UG (talhão);
              @FOR(AM(L)| DC(I,L,K) #EQ# 1: [RA] ! para cada alternativa
                     @SUM(AM(M):
                            @SUM(UG(J)| J #NE# I #AND# DC(J,M,K) #EQ# 1:
                            CORTE(J,M) * ADJ(I,J))) <= @SUM(UG(J): @SUM(AM(N) |
                            DC(J,N,K) #EQ#1: ADJ(I,J)*(1-CORTE(I,L))))
              );
       );
);

!Calcula a produção anual;
@FOR( HP(K):
       @SUM( UG(I): @SUM( AM(J): VOL(I, J, K) * CORTE(I,J) ) ) = VOL_HP(K) );

!Restrições de produção MÁXIMA e MÍNIMA anual;
@FOR( HP(K): VOL_HP(K) < VOL_MAX );
@FOR( HP(K): VOL_HP(K) > VOL_MIN );

!Restrições: somente uma alternativa de manejo por unidade de gestão;
@FOR( UG(I): @SUM( AM(J): CORTE(I,J) ) <= 1 );

!Restrião de integridade/binário;
@FOR( UG(I): @FOR( AM(J): @BIN(CORTE(I,J) )) );

END

TERSE
GO
DIVERT SOLUCAOPLI.txt
NONZ CORTE
NONZ VOL_HP
RVRT
```

APÊNDICE VI – Exemplo de agendamento com a melhor solução BT para o problema ARM_HP20 (D = desbaste, C = corte raso seguido de plantio).

Talhão	Horizonte de planejamento (anos)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1			D						C									D		
2			D					C									D			
3									C									D		
4									C										D	
5										C										
6	D							C											D	
7		D			C												D			C
8		D					C									D				
9		D				C												D		
10		D				C										D				
11	D				C												D			C
12	D				C											D				C
13	D				C											D				C
14	D				C												D			C
15	D				C												D			C
16	D				C												D			C
17		D			C												D			C
18	D				C												D			C
19		D			C												D			C
20		D			C												D			C
21	D				C											D				C
22	D						C												D	
23			D				C										D			
24			D				C										D			
25			D				C												D	
26			D				C												D	
27			D				C											D		
28	D						C											D		
29			D				C											D		
30	D					C												D		
31			D					C										D		
32			D					C									D			
33			D					C										D		
34			D					C										D		
35			D					C									D			
36			D					C										D		
37			D			C										D				
38			D				C										D			
39			D				C												D	
40			D			C												D		
41	D							C										D		
42			D				C											D		
43			D					C									D			
44	D						C												D	
45			D					C											D	
46	D						C												D	
47			D					C										D		
48			D				C									D				
49			D					C											D	
50			D				C										D			
51			D					C											D	
52			D				C											D		
53			D				C											D		
54	D						C										D			
55			D				C												D	
56	D					C											D			
57			D						C									D		
58			D			C										D				

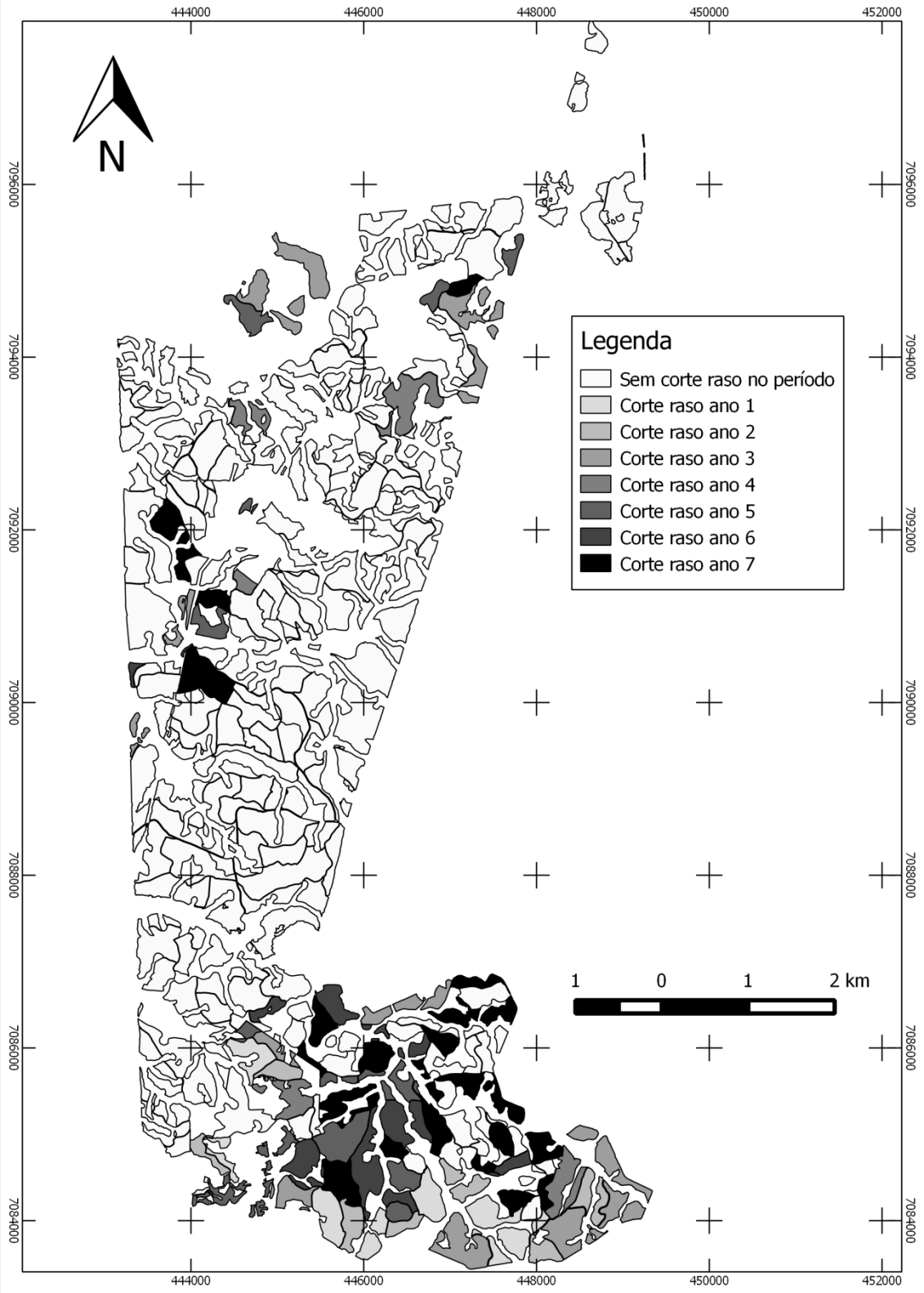
Talhão	Horizonte de planejamento (anos)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
59			D				C											D		
60			D					C									D			
61			D						C									D		
62			D				C										D			
63			D				C											D		
64				D			C										D			
65				D				C									D			
66				D						C									D	
67				D				C										D		
68				D					C									D		
69				D					C									D		
70				D				C									D			
71				D				C									D			
72				D								C								D
73				D				C										D		
74				D					C									D		
75				D				C									D			
76				D					C										D	
77				D				C										D		
78				D				C									D			
79				D				C									D			
80				D									C							
81				D			C											D		
82			D							C								D		
83					D				C										D	
84					D				C										D	
85					D				C									D		
86					D				C										D	
87					D								C							
88					D							C								D
89					D									C						
90					D								C							
91					D								C							
92					D									C						
93					D							C								D
94					D							C								D
95					D							C								D
96					D							C								D
97					D				C										D	
98					D							C								D
99					D					C									D	
100					D							C								D
101					D				C										D	
102					D					C									D	
103				D								C								D
104					C											D				C
105							C											D		
106						C												D		
107						D										C				
108						D										C				
109						D							C							
110							D								C					
111							D									C				
112							D									C				
113							D						C							
114									C										D	
115							D								C					
116							D								C					
117							D								C					
118							D						C							
119							D								C					
120							D								C					

Talhão	Horizonte de planejamento (anos)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
121								C											D	
122									C										D	
123									C										D	
124								D					C							
125								D								C				
126								D								C				
127								D									C			
128						D									C					
129								D								C				
130										C									D	
131										C									D	
132								D								C				
133								D							C					
134											C									
135											C									
136										C									D	
137										C									D	
138									D									C		
139									D						C					
140											C								D	
141									D						C					
142									D							C				
143											C								D	
144									D					C						
145									D								C			
146									D									C		
147										D							C			
148										D					C					
149										D							C			
150										D						C				
151											D				C					
152											D					C				
153											D							C		
154											D					C				
155											D						C			
156											D								C	
157											D					C				
158											D						C			
159											D					C				
160											D							C		
161											D							C		
162											D							C		
163											D						C			
164											D						C			
165											D							C		
166											D						C			
167											D					C				
168											D							C		
169											D					C				
170											D						C			
171												D							C	
172	D								C									D		
173			D									C								
174												D							C	
175		C												D						C
176	C												D						C	
177		C												D				C		
178		C												D					C	
179		C												D				C		
180		C												D			C			
181		C												D						C
182	C												D						C	

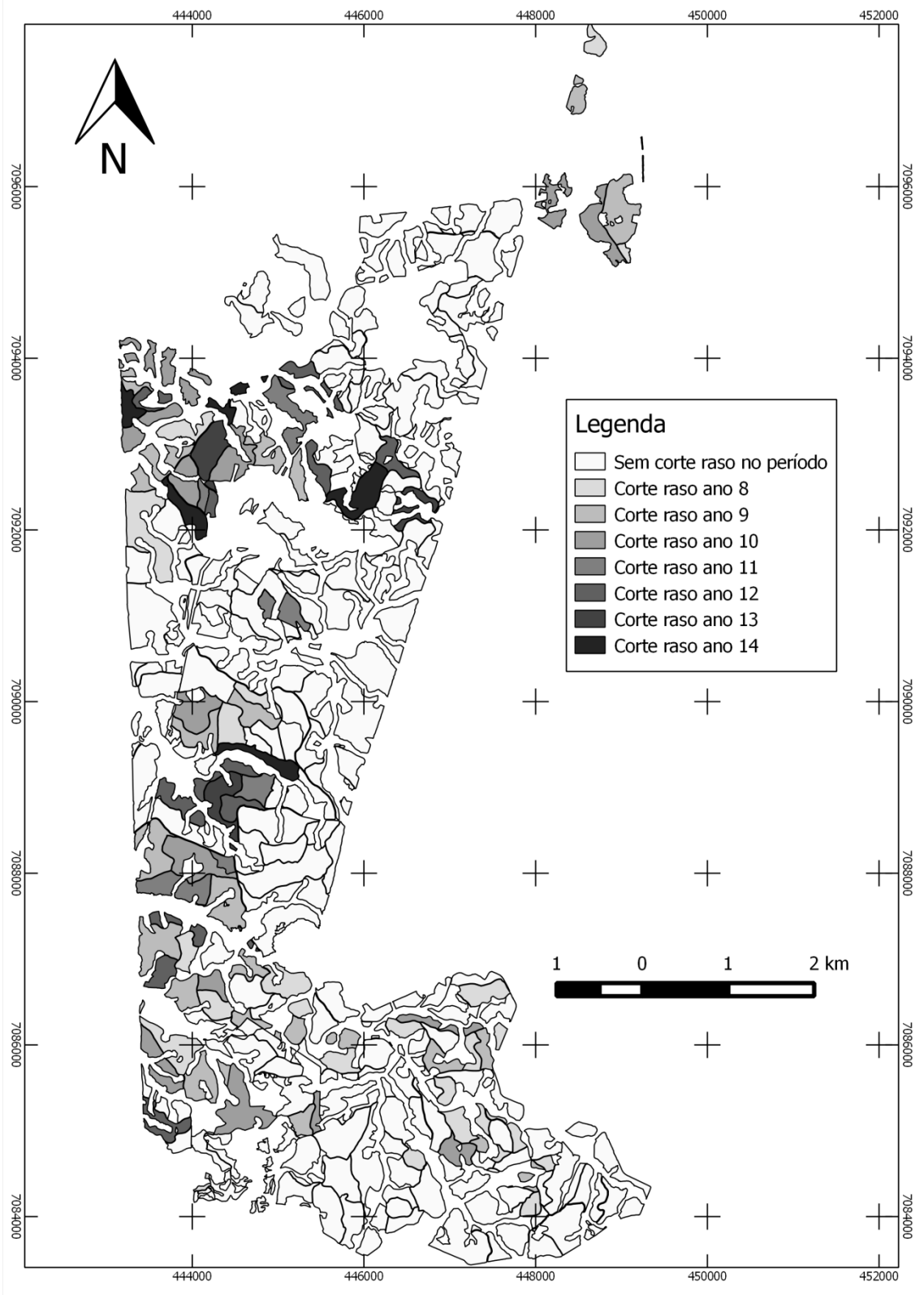
Talhão	Horizonte de planejamento (anos)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
183	C												D						C	
184	C												D				C			
185	C												D				C			
186			C												D					C
187		C												D				C		
188	C												D							C
189			C												D					C
190	C												D					C		
191		C												D				C		
192				C												D				C
193	C												D						C	
194				C												D				C
195			C												D					C
196				C												D				C
197		C											D						C	
198				C												D			C	
199			C												D				C	
200				C												D				C
201				C												D			C	
202				C												D				C
203			C												D					C
204			C												D				C	
205							C										D			
206				C												D			C	
207						D								C						
208						D								C						
209						D								C						
210						D								C						
211						D							C							
212						D						C								D
213						D						C								D
214						D								C						
215						D							C							
216						D					C									
217						D								C						
218						D									C					
219						D						C								D
220						D				C									D	
221						D									C					
222						D									C					
223						D									C					
224							D							C						
225				C												D				C
226				C												D			C	
227				C												D				C
228					C												D			C
229							C										D			
230					C											D				C
231											C								D	
232		C												D						C
233	C												D				C			
234	C												D				C			
235	C												D						C	
236		C												D						C

APÊNDICE VII – Localização dos talhões nos quais o planejamento indica o corte raso nos períodos entre 1 e 7 anos, 8 e 14 anos e 15 e 20 anos, para o problema URM_HP20 com a melhor solução do algoritmo BT.

Localização dos talhões com agenda de colheita entre os períodos 1 e 7.



Localização dos talhões com agenda de colheita entre os períodos 8 e 14.



Localização dos talhões com corte raso entre os períodos 15 e 20.