ISSN 2175-6813
Revista Engenharia na Agricultura
V.27, n.1, p.45-53, 2019
Viçosa, MG, DEA/UFV - DOI: 10.13083/reveng.v27i1.879

NOTA TÉCNICA:

UM SISTEMA PARA REGULAR A PRODUÇÃO FLORESTAL POR MEIO DA FORMAÇÃO OTIMIZADA DE UNIDADES DE PRODUÇÃO

Marcelo Otone Aguiar[1], Rodrigo Freitas Silva[2], Mayra Luiza Marques da Silva[3], Geraldo Regis Mauri[4] & Gilson Fernandes da Silva[5]

1 - Tecnólogo em Processamento de Dados, Professor Assistente da UFES/Alegre-ES, marcelo.aguiar@ufes.br
2 - Bacharel em Ciência da Computação, Professor Adjunto da UFES/Alegre-ES, rodrigo.f.silva@ufes.br
3 - Engenheira Florestal, Professora Adjunta da UFSJ/São João Del-Rei-MG, Bolsista PQ do CNPq, mayralmsilva@ufsj.edu.br
4 - Bacharel em Ciência da Computação, Professor Associado da UFES/Alegre-ES, Bolsista PQ do CNPq, grmauri@gmail.com
5 - Engenheiro Florestal, Professor Associado da UFES/Jerônimo Monteiro-ES, Bolsista PQ do CNPq, fernandes5012@gmail.com

Palavras-chave:
manejo florestal de precisão
manejo florestal sustentável
métodos heurísticos
otimização da colheita florestal

RESUMO

Um dos principais desafios atribuído ao planejamento da exploração de florestas nativas é a regulação da produção florestal, muitas vezes realizada por meio da divisão da Área de Manejo Florestal (AMF) em Unidades de Produção (UP). Levando em consideração a existência de um mercado cada vez mais competitivo e as dificuldades em alocar manualmente as UPs de maneira ótima, os manejadores vêm utilizando progressivamente sistemas computacionais para regular a produção florestal de maneira otimizada. Dentre as diversas formas de obter as soluções desse problema, destacam-se o uso dos métodos heurísticos. Sua utilização é justificada por serem métodos de solução reconhecidamente eficientes e devido à inviabilidade do uso de métodos exatos. Nesse sentido, o presente trabalho teve como objetivo o desenvolvimento de um sistema de apoio a decisão capaz de regular a floresta otimizando a alocação das UPs em uma área a ser manejada. O aplicativo foi implementado na linguagem de programação Java utilizando uma heurística construtiva gulosa e as meta-heurísticas GRASP e *Simulatted Annealing* como métodos de solução. Após a execução do sistema, o usuário pode visualizar graficamente as UPs formadas e analisar os dados das soluções obtidas por meio dos arquivos de texto gerados.

Keywords:
heuristic methods
optimization of forest harvesting
precision forestry management
sustainable forest management

A SYSTEM TO REGULATE FOREST PRODUCTION BY AN OPTIMIZED DESIGN OF PRODUCTION UNITS

ABSTRACT

Production forest regulation is one of the main challenges of the native forests exploration, often carried out through the division of the Forest Management Area (AMF) into Production Units (UP). Considering the existence of an increasingly competitive market and the difficulties in manually allocating the UPs in an optimal way, the managers have been progressively using computational systems to regulate forest production in an optimized way. Among the several ways to obtain the solutions of this problem, it might be highlighted the use of the heuristic methods. That is because they are efficient solutions' methods and due to the infeasibility of using exact methods. Thereby, the present work aims the development of a decision support system, which can regulate the forest and optimize the allocation of UPs in an area to be managed. The application was implemented in the Java programming language using a greedy constructive heuristic, and the metaheuristics GRASP and *Simulatted Annealing* as solution methods. After running the system, the user can graphically visualize the formed UPs and analyze the data of the solutions obtained through the text files generated.

Recebido para publicação em 26/04/2018 • Aprovado em 02/08/2018 • Publicado em 15/03/2019

INTRODUÇÃO

Um dos principais desafios enfrentados pelos manejadores durante o planejamento da exploração de florestas nativas é a regulação da produção florestal (CARVALHO *et al*., 2015). Tradicionalmente, a Área de Manejo Florestal (AMF) é dividida manualmente em Unidades de Produção (UP), cujos tamanhos e formas são aparentemente semelhantes sempre que possível. Entretanto, esse procedimento não corresponde necessariamente a uma produção regular, principalmente se tratando de volume de madeira e renda obtida por cada UP. Isso acontece, sobretudo, porque a heterogeneidade espacial da distribuição das espécies propicia distorções nas áreas definidas para cada UP (LEUSCHNER, 1984; SANTOS, 2012).

Considerando o horizonte de planejamento, a UP pode ser definida como Unidade de Produção Anual (UPA). A alocação física das UPAs pode ser feita, por exemplo, em função do inventário censitário, identificação de áreas produtivas ou então a partir das Áreas de Preservação Permanente (APP). Os aspectos técnicos também são importantes, como aqueles que definem a intensidade da exploração, o número de indivíduos exploráveis, alocação de pátios de estocagem e o ciclo de corte. Além disso, também podem ser consideradas informações, como o valor comercial das espécies, a distância até os clientes, os insumos necessários, os custos de produção e a tecnologia para se realizar a exploração.

A necessidade de um planejamento espacial detalhado tem sido notória nos últimos anos, devido aos fatores econômicos, ambientais e metas a serem alcançadas (BASKENT & KELES, 2005). Nesse contexto, a alocação das UPs é um problema que requer a melhor solução possível, podendo-se empregar, por exemplo, técnicas de Programação Matemática (PM) para obtenção dos resultados (SOUZA & SOARES, 2013). Contudo, o espaço de solução extremamente grande desse problema restringe o uso de algumas dessas técnicas por torná-las ineficientes. Nesse caso, a aplicação da Pesquisa Operacional (PO) se torna interessante, vem crescendo e tem apresentado relevantes contribuições para o planejamento florestal (BASKENT & KELES, 2005; MATHEY *et al*., 2008; FOTAKIS, 2015; LIU & LIN, 2015).

A fim de se chegar a resultados processados computacionalmente em tempo hábil, outros métodos de otimização devem ser investigados (SILVA *et al*., 2018). Assim, no ramo da PO, os métodos heurísticos têm se mostrado uma alternativa eficaz e eficiente para resolver os problemas de otimização, por permitirem a busca dentro do espaço de soluções sem que todas as soluções sejam avaliadas (AGUIAR *et al*., 2018).

Dentre os inúmeros métodos heurísticos existentes na literatura, há os que são promissores para a obtenção de soluções para problemas de alocação de UPs, como as meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Simulated Annealing* (SA) (MLADENOVIĆ *et al*., 2007). Esses dois métodos de solução vêm frequentemente sendo utilizados por pesquisadores em problemas que possuem elevada complexidade de tempo de execução e são considerados difíceis de serem resolvidos computacionalmente, se enquadrando em uma classe geral de problemas conhecida em complexidade de algoritmos como NP-difícil (CORMEN *et al*., 2012).

Este trabalho teve como objetivo a implementação de um sistema para regular a floresta em termos de volume de madeira e renda obtidos a partir da otimização da alocação das UPAs. A ideia é que cada unidade seja explorada anualmente e tenham produções volumétricas de madeira e receitas semelhantes. Para isso, foi utilizada uma heurística construtiva gulosa e as meta-heurísticas GRASP e SA. Além disso, a aplicação construída conta com uma interface gráfica intuitiva, tornando simples a configuração dos dados de entrada, a parametrização dos métodos de solução e a interpretação dos resultados.

MATERIAL E MÉTODOS

Inicialmente, foi implementada uma heurística construtiva gulosa para obter uma formação inicial para as UPAs, feita de maneira rápida e aleatória (ZIVIANI, 2013). Posteriormente, foi implementado o algoritmo da meta-heurística GRASP (FEO & RESENDE, 1989) e o SA (KIRKPATRICK *et al*., 1983), com o objetivo de melhorar os resultados encontrados inicialmente. A linguagem de programação utilizada foi o Java (DEITEL, 2010), juntamente com um ambiente integrado de desenvolvimento, o Netbeans (DANTAS, 2011).

O problema de alocação das UPAs tem as mesmas características do problema clássico de localização-alocação, também conhecido em otimização combinatória, como problema das p-medianas. O objetivo é formar as UPs de acordo com a

distribuição espacial das árvores inventariadas, levando em conta a regulação da floresta. Os dados de entrada no sistema correspondem às árvores inventariadas com suas respectivas localizações e estimativas de volume e de renda. Neste caso, foi utilizada a equação da distância euclidiana (Equação 1) para calcular a Função Objetivo (FO) do problema (Equação 2), seguindo os passos de Silva *et al*. (2018). A melhor solução é aquela que resulta em uma menor distância média entre as árvores de uma UPA, fazendo com que todas as UPAs tenham receitas e volumes de madeira semelhantes. Foi necessário construir uma matriz simétrica para armazenar, previamente, a distância entre todas as árvores contidas na base de dados. Essa operação é executada durante a importação dos dados. Por fim, a função implementada para calcular a FO foi utilizada em todas as heurísticas e meta-heurísticas.

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (1)$$

$$\mathrm{FO}_{\min} = \frac{\sum_{i=1}^{n} d_{i,j}}{n} \qquad \forall j \quad (2)$$

em que,
$d_{i,j}$=distância euclidiana entre as árvores i e j;
x_i =posição no eixo x da árvore i;
x_j =posição no eixo x da árvore j;
y_i =posição no eixo y da árvore i;
y_j =posição no eixo y da árvore j, e
$\mathrm{FO}_{\min}$ = menor FO calculada;
n = número de árvores.

A estratégia gulosa é tipicamente empregada na literatura para resolver problemas de otimização de uma forma simples e rápida, sem garantias, contudo, de se chegar à solução ótima. É uma técnica útil para a resolução de muitos problemas, sem que haja a necessidade de utilizar métodos mais robustos. Suas vantagens são: facilidade de implementação e eficiência, o que justificam seu uso. Um algoritmo guloso sempre faz a escolha que parece ser a melhor no momento, ou seja, faz uma escolha ótima para as condições locais. A propriedade de escolha gulosa garante que uma sequência de escolhas locais ótimas pode levar a uma solução global ótima para o problema (CORMEN *et al*., 2012). As escolhas feitas nunca são reavaliadas, ou seja, não há qualquer tipo de *backtracking*. Portanto, em cada passo da resolução do problema, a escolha deve ser possível, localmente ótima e irreversível.

A heurística construtiva gulosa foi implementada da seguinte maneira: primeiramente, escolhe-se a quantidade de UPAs que deverão ser formadas. Cada UPA terá um ponto central (localização geográfica) selecionado aleatoriamente na área de estudo. A partir daí, de forma gulosa, as árvores inventariadas vão se agrupando ao ponto central mais próximo, formando uma espécie de *clusters* que, ao final do processo de agrupamento, constituirá as UPAs. Esse tipo de alocação leva prioritariamente em consideração a menor distância entre as árvores e o ponto central das UPAs. O passo seguinte é o cálculo da função objetivo (FO) do problema com base nos *clusters* formados e, por último, a validação da solução. A solução só é válida se as UPAs formadas estiverem reguladas em termos de volume e renda, caso não estejam, a solução é rejeitada.

O algoritmo guloso é executado *n* vezes (iterações), alcançando em cada execução uma solução potencialmente diferente. As soluções encontradas são comparadas entre si, armazenando sempre aquela que possui a menor FO. As demais soluções são simplesmente descartadas. A Figura 1 descreve o código base da heurística construtiva gulosa.

```
1  mS ← ∅
2  mFO ← ∞
3  enquanto condição de parada faça
4      aS ← selecionar(candidatos)
5      aFO ← calculaFO(aS)
6      se aFO < mFO então
7          mS ← aS
8          mFO ← aFO
9      fimse
10 fimenquanto
11 retorna mS
```

Fonte: Adaptado de Ziviani (2013, p. 59).

Figura 1. Código base da heurística construtiva gulosa.

A meta-heurística GRASP consiste em um procedimento iterativo de duas fases para criar uma solução inicial e depois efetuar uma busca local para melhorar a qualidade da solução gerada. A solução inicial é baseada nas três primeiras

iniciais de sua sigla em inglês: gulosa (*Greedy*), aleatória (*Randomized*) e adaptativa (*Adaptive*). O GRASP procura utilizar as melhores características dos algoritmos gulosos e procedimentos aleatórios para a construção das soluções iniciais. Essas soluções, posteriormente, são refinadas por meio da segunda fase desse algoritmo, baseada em buscas locais (SILVA *et al.*, 2015). As buscas locais são basicamente responsáveis por tornar válida as soluções não reguladas, melhorando, assim, as soluções existentes.

Neste trabalho, foram implementadas duas estratégias para a busca local: primeira melhora e melhor melhora. A melhor melhora escolhe em cada etapa uma das posições de busca na vizinhança que resulte em uma melhoria máxima possível para o valor da FO. A primeira melhora funciona de forma similar, a única diferença é que, ao encontrar a primeira melhor solução, o processo é interrompido (AGUIAR *et al*., 2018). A fase de construção da solução inicial também permite a adoção de várias estratégias. Entretanto, a mais conhecida é a semi-gulosa proposta por Hart e Shogan (1987). O algoritmo base do GRASP é apresentado na Figura 2.

```
1   mS ← ∅
2   mFO ← ∞
3   para i de 1 até max_iter faça
4       aS ← geraSolucao(LRC)
5       aS ← buscaLocal(aS)
6       aFO ← calculaFO(aS)
7       se aFO < mFO então
8           mS ← aS
9           mFO ← aFO
10      fimse
11  fimpara
12  retorna mS
```

Fonte: Adaptado de Glover e Kochenberger (2003, p. 220).

Figura 2. Código base da meta-heurística GRASP.

Como a fase de construção envolve gulosidade, os problemas permitem a ordenação dos elementos de forma a possibilitar a obtenção do item de "maior valor" associado. Contudo, no GRASP, não necessariamente isso irá ocorrer, pois esta etapa é flexibilizada por meio de uma Lista Restrita de Candidatos (LRC). Assim, apenas os candidatos pertencentes a esta lista irão compor a solução na fase inicial. A LRC é um parâmetro do GRASP que determinará quantos candidatos vão fazer parte da lista (GOLDBARG *et al*., 2016). A Tabela 1 apresenta os parâmetros do GRASP.

Tabela 1. Parâmetros do GRASP.

Parâmetro	Descrição
max_iter	Número de iterações
LRC	Lista restrita de candidatos

Fonte: Adaptado de Aguiar *et al*. (2018, p. 146).

No caso da meta-heurística SA, a inspiração tem origem na área físico-química. Busca-se mimetizar o recozimento de materiais como metais ou vidros (GOLDBARG *et al*., 2016; MENDES *et al.*, 2017). De acordo com Goldbarg *et al*. (2016), o SA baseia a sua estratégia principalmente em três características: 1) Uso de uma solução inicial; 2) Geração aleatória de novas soluções com base na solução corrente e 3) Critério progressivamente elitista para trocar a solução aleatória pela atual.

O algoritmo base do SA é apresentado na Figura 3.

```
1   S* ← S
2   IterT ← 0
3   T ← T0
4   enquanto T > Tc faça
5       enquanto IterT < SAmax faça
6           IterT ← IterT + 1
7           gerar(um vizinho S' ∈ N(S))
8           Δ ← f(S') - f(S)
9           se Δ < 0 então
10              S ← S'
11              se f(S') < f(S*) então
12                  S* ← S'
13              fimse
14          senão
15              tomar(x ∈ [0,1])
16              se x < e^(-Δ/T) então
17                  S ← S'
18              fimse
19          fimse
20      fimenqanto
21      T ← α x T
22      IterT ← 0
23  fimenqanto
24  retorna S*
```

Fonte: Adaptado de Goldbarg *et al*. (2016, p. 110).

Figura 3. Código base da meta-heurística SA.

O SA recebe como argumentos de entrada a taxa de resfriamento, o número de iterações, a temperatura inicial, a temperatura de congelamento e a solução inicial, conforme apresentado na Tabela 2. Após a parametrização do algoritmo e a execução do mesmo, as UPAs vão sendo formadas e as soluções vão sendo progressivamente melhoradas até encontrar os formações reguladas com a menor distância média possível entre as árvores.

Tabela 2. Parâmetros do AS.

Parâmetro	Descrição
α	Taxa de resfriamento
SA_{max}	Número de iterações
T_o	Temperatura inicial
T_c	Temperatura de congelamento
S	Solução inicial

Fonte: Adaptado de Aguiar *et al*. (2018, p. 146).

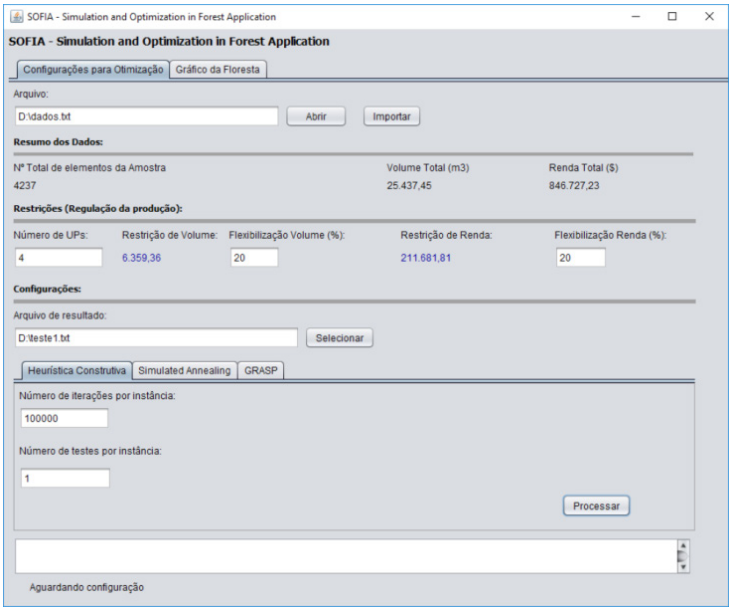
Os primeiros passos do algoritmo são a inicialização da solução global com a solução inicial, a inicialização da variável $IterT$, responsável pela contabilização das iterações, e a inicialização da temperatura de controle. O primeiro laço do código (linha 4 da Figura 3) mantém as iterações ativas, enquanto a temperatura inicial não atinge a temperatura de congelamento. O restante da codificação é responsável por explorar o espaço de vizinhança da solução inicial.

RESULTADOS E DISCUSSÃO

A aplicação desenvolvida foi denominada SOFIA (*Simulation and Optimization in Forest Application*) e disponibiliza uma interface simples e intuitiva, conforme pode ser visualizada na Figura 4. São vários os parâmetros de entrada, o primeiro é o arquivo de origem que deve ser selecionado pelo botão abrir e, após a escolha do arquivo, os dados devem ser importados para a memória com o botão importar. Um resumo dos dados é exibido a seguir com o total de elementos, o volume total em metros cúbicos e a renda total, referente à base de dados selecionada.

Outros parâmetros de entrada são: o número de UPs que se pretende formar, a flexibilização para a restrição de volume e a flexibilização para a restrição de renda. Ambas as restrições precisam ser relaxadas por meio de flexibilizações, pois não seria prático forçar valores exatos para a divisão das UPs. A flexibilização indica, por exemplo, em até quantos por cento uma UP pode se distanciar das demais em termos de renda e volume de madeira explorada. Trata-se, portanto, da regulação da unidades de trabalho.

Por fim, é necessário informar o local e o nome do arquivo para gravar os resultados da execução



Fonte: Os autores.

Figura 4. Tela inicial da aplicação SOFIA.

dos métodos heurísticos. Após configurar estes parâmetros que são gerais, deve-se escolher entre a heurística construtiva, o SA e o GRASP. Na Figura 4, foi selecionada a heurística construtiva. Neste caso, observe que estão disponíveis dois argumentos de entrada: o número de iterações, que determinará o número de vezes que a heurística será executada em busca de uma solução com melhor FO, e o número de testes por instância, que define quantos testes de *n* iterações por instância de dados devem ser executados.

Ao selecionar a aba do SA, conforme é mostrado na Figura 5, são disponibilizados os seus parâmetros de configuração, seguindo as informações disponíveis na Tabela 2, e outros três parâmetros que possibilitam customizar essa meta-heurística. O primeiro parâmetro, dentre os três, permite determinar como a solução inicial será gerada. Por padrão, a solução inicial é gerada de forma aleatória, sem garantia de que tal solução seja viável. Isso pode ser um problema, pois como o SA tem a estratégia de intensificar a busca a partir da solução inicial, se esta não é viável, então o espaço de soluções que o SA irá percorrer pode não ser promissor. Como consequência, os resultados não serão bons. Por isso, também é disponibilizada a opção de gerar a solução inicial através da heurística construtiva gulosa.

Similarmente à heurística construtiva, para o SA também é possível configurar o número de testes por instância que devem ser executados. Outro parâmetro que deve ser configurado é o peso para a penalização das soluções ruins. Diferentemente da estratégia gulosa, em que soluções inviáveis são rejeitadas, no SA, rejeitar uma solução poderia diminuir muito o seu desempenho ao buscar uma solução vizinha. Dessa forma, optou-se por penalizar as soluções inviáveis em que ω, ω, refere-se ao parâmetro correspondente ao peso da penalização que deve ser configurado. Assim, se a solução for inviável, violando alguma das restrições, então a diferença entre o valor excedido que levou à inviabilidade e o valor limite será multiplicada por ω e somado ao valor da FO, conforme é mostrado na Equação 3.

$$R_\omega P: Min\ cx + \omega(\max(0, e - D_x)) \quad (3)$$

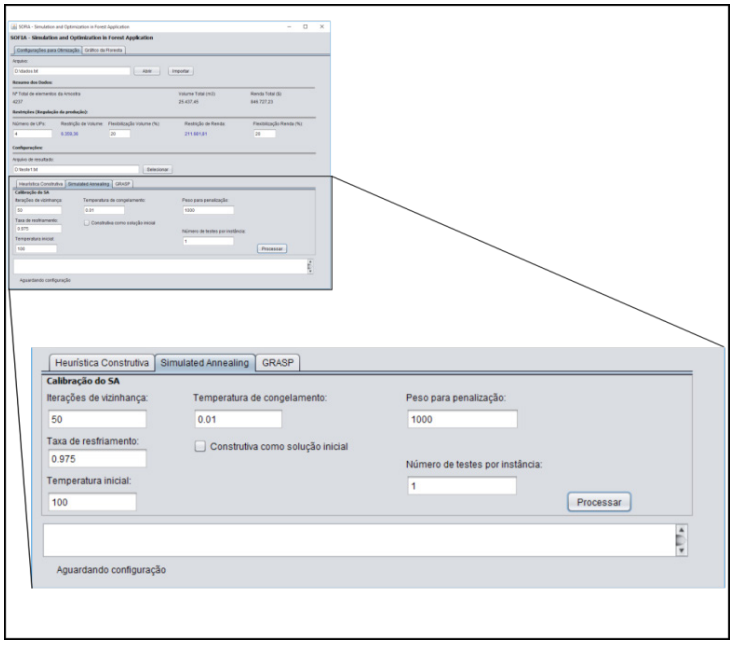
em que,

$R_\omega P: Min\ cx$ = penalização da função objetivo associada ao problema;

ω = fator de grandeza para a penalização;

e = valor obtido para a restrição do problema; e

D_x = valor limite associado à restrição do problema.



Fonte: Os autores.

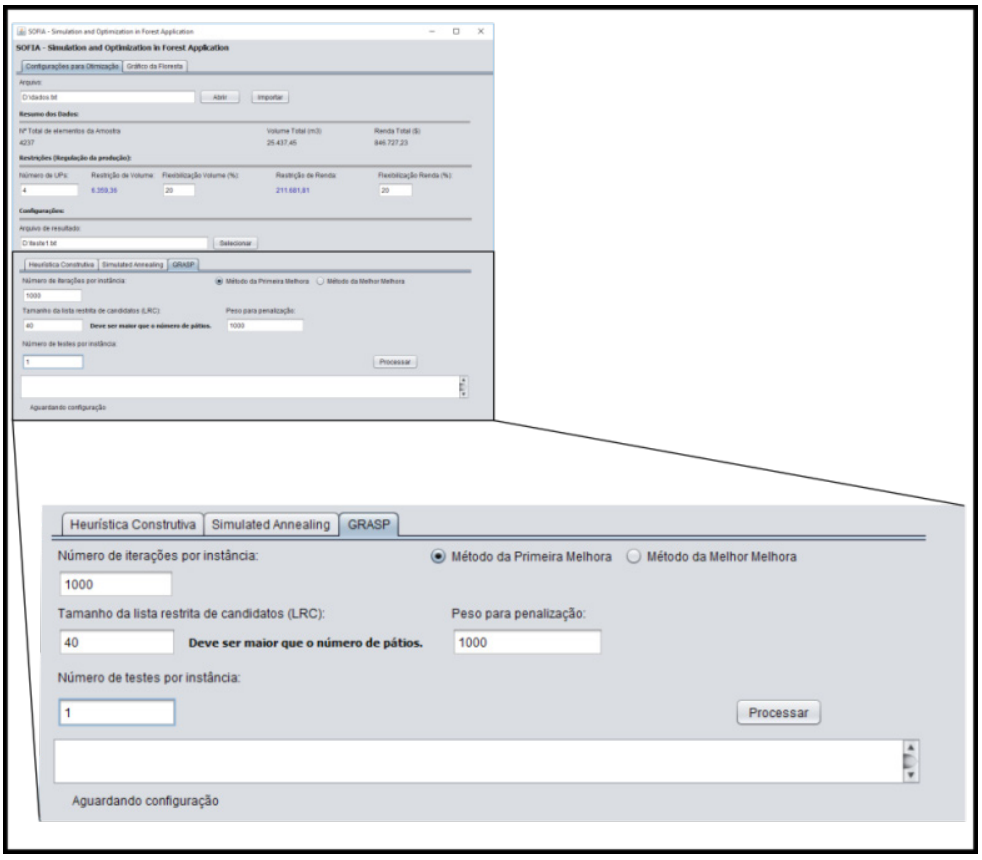
Figura 5. Configuração do SA.

A aba relacionada à meta-heurística GRASP (Figura 6) disponibiliza ao usuário os parâmetros de configuração desse algoritmo, descritos na Tabela 1, de forma análoga aos demais métodos já mencionados anteriormente. Além disso, também são disponibilizados alguns parâmetros que permitem customizar a execução dessa meta-heurística, como a escolha do algoritmo de busca local. Similarmente ao SA, o GRASP também emprega a penalização das soluções durante o processo de busca, podendo esta ocorrer tanto na etapa de construção da solução inicial, como na etapa de busca local.

Cabe destacar a configuração da busca local do GRASP, podendo ser utilizado o método da primeira melhora ou a melhor melhora. Embora as duas técnicas tenham um modo de operação similar, o método da primeira melhora, no geral, possui um desempenho melhor, em termos de tempo de processamento, visto que o procedimento é interrompido ao encontrar a primeira solução com melhor valor de FO.

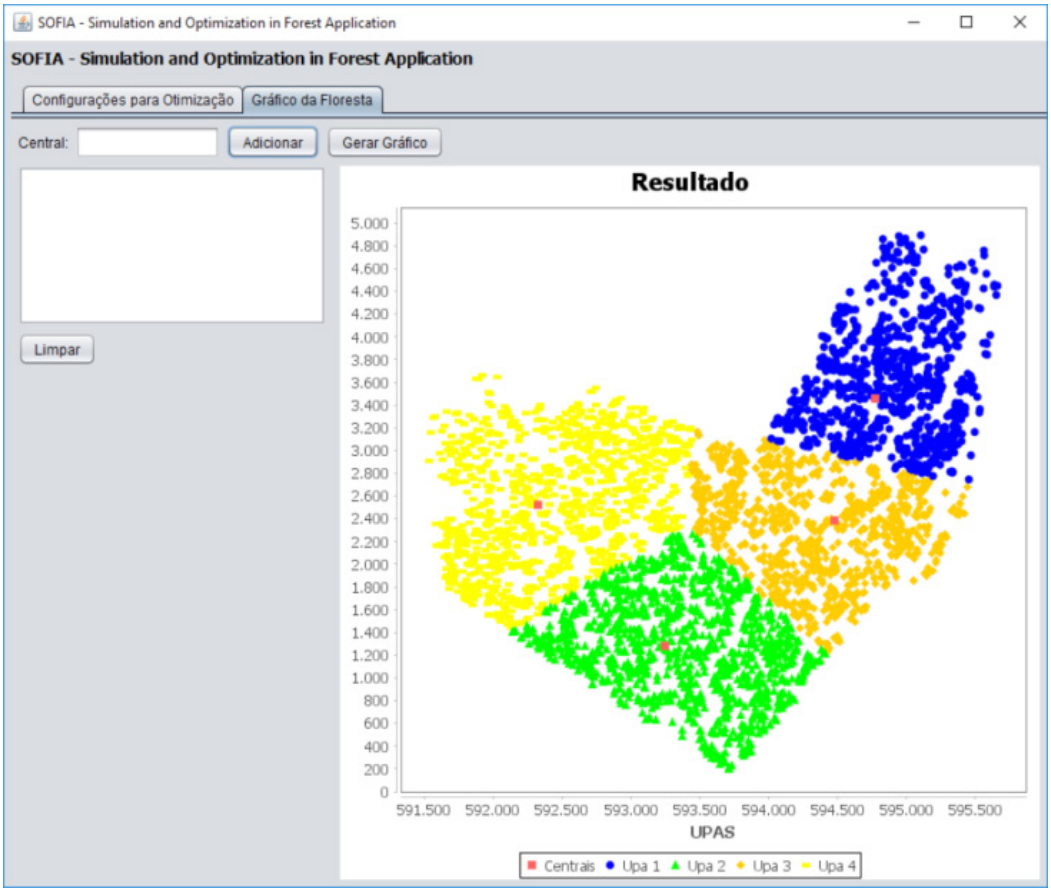
Outro recurso disponível no aplicativo é a geração do gráfico com os indivíduos da floresta em análise. O gráfico permite que o manejador florestal visualize a formação espacial das UPAs e as respectivas árvores associadas a cada uma delas, após a execução dos algoritmos de otimização. A Figura 7 ilustra a melhor solução encontrada para uma área florestal experimental regulada e subdividia em 4 UPAs. Os pontos demarcados de vermelho são as localizações centrais de cada UPA.

Quando o manejador executa apenas um teste, o gráfico é gerado automaticamente para a solução com a melhor FO obtida pelo método heurístico selecionado. Se o sistema for configurado para executar um número de testes superior a um, do tipo em "lote", o gráfico não será gerado automaticamente, mas os resultados serão disponibilizados em arquivo texto com todas as informações das soluções encontradas. A partir desse arquivo, é possível gerar a ilustração da melhor solução encontrada pressionando o botão adicionar.



Fonte: Os autores.

Figura 6. Configuração do GRASP.



Fonte: Autores.

Figura 7. Tela para visualização do gráfico.

CONCLUSÕES

- Este trabalho abordou a otimização da regulação de UPs por meio do problema de localização-alocação com a utilização de métodos heurísticos.
- O trabalho apresenta um sistema computacional, o SOFIA, contendo uma metodologia de otimização baseada nos seguintes métodos: heurística construtiva gulosa, SA e GRASP.
- Também são apresentadas as interfaces gráficas implementadas e as informações necessários para utilização dessa ferramenta.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, M.O.; MAURI, G.R.; SILVA, R.F. **Introdução aos Métodos Heurísticos de Otimização com Python.** 1 ed. Alegre, ES: CAUFES, 2018.

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

CARVALHO, K.H.A.; SILVA, M.L.; LEITE, H.G.; BINOTI, D.H.B. Influência da taxa de juros e do preço da madeira em modelos de regulação florestal. **Pesquisa Florestal Brasileira**, Colombo, v.35, n.82, p.143-151, 2015.

CORMEN, T.H. et al. **Algoritmos:** teoria e prática. 3 ed. [s.l.]: Elsevier, 2012. 944p.

DANTAS, R. **NetBeans IDE 7 Cookbook.** Packt Publishing, 2011.

DEITEL, H. **Java: como programar.** Prentice Hall Brasil, 2010.

FEO, T.A.; RESENDE, M.G.A probabilistic

heuristic for a computationally difficult set covering problem. **Operations Research Letters**, v.8, n.2, p.67-71, 1989.

FOTAKIS, D.G. Multi-objective spatial forest planning using self-organization. **Ecological Informatics**, v.29, p.1-5, 2015.

GOLDBARG, M.C.; GOLDBARG, E.G.; LOUREIRO, H.P. **Otimização combinatória e meta-heurísticas:** Algoritmos e aplicações. 1 ed. Rio de Janeiro: Elsevier, 2016.

HART, J.; SHOGAN, A.W. Semi-greedy heuristics: An empirical study. **Operations Research Letters**, v.6, n.3, p.107-114, 1987.

KIRKPATRICK, S.; GELATT, C.D.; VECCHI, M.P. Optimization by simulated annealing. **Science**, v.220, n.4598, p.671-680, 1983.

LEUSCHNER, W.A. **Introduction to forest resources management**. New York: Jonh Willey & Sons, 1984. 298p.

LIU, W.; LIN, C. Spatial forest resource planning using a cultural algorithm with problem-specific information. **Environmental Modelling & Software**, Amsterdam, v.71, p.126-137, 2015.

MATHEY, A.H.; KRCMAR, E.; DRAGICEVIC, S.; VERTINSKY, I. An object-oriented cellular automata model for forest planning problems. **Ecological Modelling**, Amsterdam, v.212, n.3-4, p.359-371, 2008.

MENDES, H.A.M., CECÍLIO, R.A.; ZANETTI, S.S. Ferramenta para calibração automática do modelo hidrológico DHSVM. **Revista Engenharia na Agricultura,** Viçosa, v.25, n.3, p.272-282, 2017.

MLADENOVIĆ, N.; BRIMBERG, J.; HANSEN, P.; MORENO-PÉREZ, J.A. The p-median problem: A survey of metaheuristic approaches, **European Journal of Operational Research**, v.179, 3ed., p.927-939, 2007.

SANTOS, A.L. **Uso da programação linear na identificação de estratégias ótimas de regulação florestal considerando mix de consumo**. 2012. 90f. Dissertação (Mestrado em Ciências Florestais) – Universidade Federal do Paraná, Curitiba, 2012.

SILVA, R.F.; MONTES, D.P.; SILVA, G.F. Aplicação da meta-heurística GRASP para o problema do sortimento florestal. In: Congresso brasileiro de ciência e tecnologia da madeira, 2., 2015, Belo Horizonte. **Anais**... Belo Horizonte: SBCTEM, 2015.

SILVA, E.F.; SILVA, R.F.; VIEIRA, G.C.; LEITE, C.C.C.; AGUIAR, M.O.; FIGUEIREDO, E.Ó.; SILVA, M.L.M.; SILVA, G.F. Planning of production units for native forest management areas in the Amazon, **Revista Brasileira de Ciências Agrárias**, Recife, v.13, n.1, p.1-8, 2018.

SOUZA, A.L.; SOARES, C.P.B. **Floresta nativas:** estrutura, dinâmica e manejo. 1ed. Viçosa, MG: UFV, 2013.

ZIVIANI, N. **Projeto de algoritmos:** com implementações em Pascal e C. 3ed. São Paulo: Cengage Learning, 2013.