OTIMIZAÇÃO HEURÍSTICA PARA ENTREGAS EFICIENTES DE SOFTWARE[1]

Luíza Nürnberg [2], Marcelo de Souza [3]

[1] Vinculado ao projeto intitulado "Projeto automático de algoritmos".
[2] Acadêmico(a) do Curso de Eng. de Software – CEAVI – Bolsista PIVIC.
[3] Orientador(a), Departamento de Eng. de Software – CEAVI – marcelo.desouza@udesc.br.

A aplicação de algoritmos heurísticos é necessária para a solução de diversos problemas NP-Difíceis, uma vez que métodos exatos podem se tornar impraticáveis por conta do elevado tempo computacional que comumente se faz necessário. Algoritmos heurísticos são capazes de produzir boas soluções em tempo razoável, mas não garantem otimalidade. Em muitos casos, espera-se que essas heurísticas alcancem soluções próximas aos valores ótimos na resolução de problemas complexos, especialmente quando partem de uma solução viável (Goldbarg, 2015).

Nesse contexto, destaca-se o problema do próximo lançamento (Bagnall, Rayward-Smith e Whittley, 2001), com aplicações relevantes na área da engenharia de software. O problema tem por objetivo aprimorar a definição do escopo e implementação da próxima versão de um software, abordando o desafio recorrente de melhoria das funcionalidades dos projetos. Uma estimativa de custos eficiente pode auxiliar no melhor ciclo de vida do software, e sua carência acarreta sérios problemas para as empresas, conforme alertado por Harman, Mansouri e Zhang (2009).

A Figura 1 ilustra o problema do próximo lançamento. Dado o conjunto de clientes da empresa (retângulos maiores, em cinza) e o conjunto de requisitos solicitados por cada cliente (retângulos menores, em verde), deve-se determinar quais clientes serão atendidos no próximo lançamento. Para que um cliente seja atendido, todos os requisitos solicitados por ele devem ser implementados. Cada requisito possui um custo de implementação, e o próximo lançamento tem um orçamento que não pode ser ultrapassado. Além disso, os requisitos possuem dependências entre si. Um requisito só pode ser implementado se todos os requisitos dos quais ele depende também forem implementados. Cada cliente possui uma importância para a empresa. Logo, deve-se escolher os clientes a serem atendidos, de tal forma que a importância total desses clientes seja maximizada.

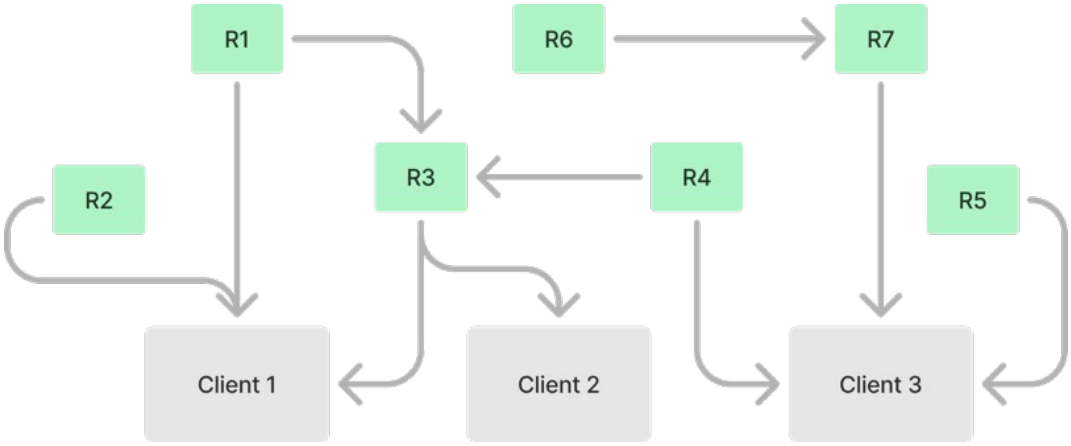

Figura 1. *Representação gráfica de uma instância do problema do próximo lançamento, incluindo os clientes, requisitos e seus relacionamentos.*

A fim de oferecer soluções automatizadas que auxiliem empresas em estratégias de atendimento às demandas dos clientes e alocação de recursos, este trabalho propõe algoritmos heurísticos de otimização para a solução do problema do próximo lançamento. Em particular, são exploradas diferentes estratégias na construção de um algoritmo guloso iterado (Ruiz, et al., 2007). Tal abordagem visa possibilitar a solução de instâncias grandes do problema, cuja solução exata tende a exigir um tempo muito elevado, devido à natureza combinatória do problema.

O algoritmo guloso iterado explora o espaço de soluções por repetidamente aplicar duas heurísticas principais: a destruição parcial da solução atual, seguida da sua reconstrução, produzindo uma (potencialmente) nova solução completa. O Algoritmo 1 apresenta o algoritmo guloso iterado proposto. O primeiro passo consiste em construir uma solução inicial. Essa construção pode ser aleatória, onde os clientes são escolhidos aleatoriamente enquanto há orçamento disponível, ou usando uma heurística construtiva semi-gulosa. Essa heurística ranqueia os clientes conforme uma função heurística, e escolhe um dos α primeiros clientes com probabilidade uniforme. Esse processo é repetido enquanto há orçamento disponível. Como funções heurísticas, podem ser usados o valor de importância do cliente, o custo total dos requisitos necessários para atender o cliente, ou a importância dividida pelo custo.

Algoritmo 1. Algoritmo guloso iterado.

```
S ← Construção(α)
S* ← S
para i ← 1 até N faça
        se estagnação e reinício ativado então
                S ← Construção(α)
        fim se
        S’ ← Destruição(S, d)
        S’ ← Construção(S’, α)
        se S’ é melhor que S* então
                S* ← S’
        fim se
        S ← Aceitação(S’, S*)
fim para
retorna S*
```

Por um número predeterminado de iterações, o algoritmo destrói parcialmente a solução e reconstrói usando a heurística construtiva semi-gulosa. Na destruição, um percentual dos clientes (definido pelo parâmetro d) é removido. A escolha dos clientes a serem removidos é feita aleatoriamente. Além disso, pode ser usada uma estratégia de reinício quando detectada estagnação da busca. Neste caso, uma nova solução é construída, seja pela construção aleatória ou pela semi-gulosa. A estagnação é detectada quando a solução incumbente não melhora por um determinado número de iterações, definido pelo parâmetro $M \in [0, 1]$ (o limiar para estagnação é definido por $M \cdot N$).

O algoritmo guloso iterado proposto possui vários componentes e parâmetros. Como componentes, ele implementa duas estratégias construtivas, duas estratégias de reinício e dois

critérios de aceitação. Como parâmetros, a aleatoriedade da construção é definida pelo parâmetro α, o tamanho da destruição é dado pelo parâmetro *d*, e a detecção de estagnação depende do parâmetro *M*. Além disso, o parâmetro *H* determina qual heurística a ser usada na construção semi-gulosa.

Para determinar o algoritmo final, i.e. quais componentes usar e quais valores atribuir aos parâmetros, diferentes versões do algoritmo devem ser avaliadas para escolher aquela de melhor desempenho. Esse processo é oneroso e suscetível a erros, como ignorar versões promissoras ou fazer uma comparação ineficaz. Em vez de fazer essa avaliação manualmente, foi adotado o configurador irace (López-Ibáñez et al., 2016), que automatiza esse processo e busca pela configuração de melhor desempenho. Foi usado um conjunto independente de instâncias de treinamento, e executado o irace com um limite de 1000 experimentos. A configuração obtida usa construção aleatória para a solução inicial, construção semi-gulosa para o reinício (que é habilitado) e para a reconstrução da solução parcial com a heurística definida pela importância do cliente dividida pelo custo total a ele associado, e usa a solução incumbente para continuar a busca na próxima iteração (aceitação). Os melhores valores de parâmetros identificados pelo irace foram $\alpha = 0{,}11$, $d = 0{,}38$ e $M = 0{,}5$.

O algoritmo guloso iterado sob a configuração produzida pelo irace foi avaliado em um conjunto de instâncias de teste comumente usado na literatura. Os experimentos foram executados com cinco replicações e usando 1000 iterações. A Tabela 1 apresenta as instâncias, os valores das suas soluções ótimas e os resultados médios obtidos pelo algoritmo heurístico.

Tabela 1. *Resultados do algoritmo guloso iterado em comparação com as soluções ótimas.*

Instância	Coeficiente	Solução ótima	Solução heurística
nrp1	0,3	1.204	1.185,4
nrp1	0,5	1.840	1.823,4
nrp1	0,7	2.507	2.507,0
nrp2	0,3	4.970	4.840,6
nrp2	0,5	8.065	7.959,0
nrp2	0,7	11316	11084,6
nrp3	0,3	7.488	7.398,0
nrp3	0,5	11.159	11.086,0
nrp3	0,7	14.196	14.188,6
nrp4	0,3	10.690	10.574,3
nrp4	0,5	15.985	15.877,7
nrp4	0,7	20.913	20.892,1
nrp5	0,3	18.510	17.912,8
nrp5	0,5	24.701	24.276,2
nrp5	0,7	28.912	28.871,6

Com exceção de uma instância, os valores médios das cinco replicações do algoritmo guloso são menores que os valores das soluções ótimas. No entanto, observa-se que o algoritmo é capaz de produzir soluções que se aproximam das soluções ótimas, o que enfatiza o desempenho satisfatório da abordagem heurística. A abordagem automatizada para determinar a versão final

do algoritmo se mostrou eficaz na produção de algoritmos de bom desempenho, bem como reduz o esforço humano na busca de configurações adequadas para o algoritmo.

Como trabalhos futuros, deseja-se estender as heurísticas implementadas, incorporando novas estratégias heurísticas, como diferentes construções, um método de destruição guiada e um procedimento de busca local para refinamento das soluções produzidas a cada iteração. Além disso, deseja-se incorporar o algoritmo guloso iterado em um software capaz de auxiliar na tomada de decisão para gestores de projetos de software.

Palavras-chave: Problema do próximo lançamento. Otimização. Algoritmos heurísticos.

BAGNALL, Anthony J.; RAYWARD-SMITH, Victor J.; WHITTLEY, Ian M. The next release problem. Information and Software Technology, 2001.

GOLDBARG, Elizabeth. Otimização Combinatória e Meta-heurísticas - Algoritmos e Aplicações. Rio de Janeiro: Grupo GEN, 2015. E-book. ISBN 9788595154667. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788595154667/. Acesso em: 10 set. 2024.

HARMAN, Mark; MANSOURI, S. Afshin; ZHANG, Yuanyuan. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Department of Computer Science, Kings College London, Tech. Rep. 2009.

LÓPEZ-IBÁÑEZ, Manuel et al. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, v. 3, p. 43-58, 2016.

RUIZ, Rubén; STÜTZLE, Thomas. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, 2007.