https://eventos.utfpr.edu.br//sicite/sicite2018

Configuração automática de metaheurísticas para otimização combinatória

Automatic configuration of metaheuristics for combinatorial optimization

Eduardo Kirsten Otte
eduardo@otte.net.br
Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil

Myriam Regattieri De Biase Da Silva Delgado
myriamdelg@gmail.com
Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil

Lucas Marcondes Pavelski
lmpavelski@yahoo.com.br
Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil

RESUMO

Meta-heurísticas (MH) e heurísticas (H) apresentam um elevado grau de sucesso na solução de problemas de otimização. Um fator que pode influenciar a qualidade da otimização é a definição dos valores dos parâmetros do otimizador sendo considerado. A abordagem proposta neste trabalho, denominada PG-t*uner*, é baseada na MH denominada programação genética (PG) e realiza uma busca no espaço dos parâmetros de uma MH (*Iterated Local Search*) de forma a otimizar o desempenho desta na solução de um determinado problema base (*flowshop*). Para isso, o PG-*tuner* trata cada possível configuração do ILS como um indivíduo cuja estrutura em árvore respeita a hierarquia dos parâmetros considerados. O conjunto de configurações tem tamanho fixo e evolui por G gerações. O *fitness* de cada configuração do ILS é computado com base no seu desempenho médio para uma instância do *flowshop*. Os resultados mostraram que a proposta é comparável a técnicas do estado da arte como o *Irace*.

PALAVRAS-CHAVE: Ajuste de parâmetros. Metaheurísticas. *Flowshop*.

ABSTRACT

Meta-heuristics (MH) and Heuristics (H) show a high rate of success in solving optimization problems. A factor that might influence the optimization quality is the definition of the considered optimizer parameter values. This work proposed approach, named GP-tuner, is based on the MH called genetic programming (GP) and performs a search in the space of parameters of an MH (Iterated Local Search, ILS) in order to optimize its performance on solving a given base problem (flowshop). A fixed number of ILS configurations evolves for G generations. Each ILS configuration fitness is computed as the mean performance on a given flowshop instance. The results show that the proposed approach is comparable to state-of-the-art techniques like Irace.

KEYWORDS: Parameter tuning. Metaheuristics. Flowshop.

Recebido: 31 ago 2018
Aprovado: 04 out 2018

INTRODUÇÃO

Problemas de otimização consistem em encontrar o melhor valor (máximo ou mínimo) de uma função objetivo. Estes problemas surgem em diversas áreas do conhecimento e muitas vezes não podem ser resolvidos por algoritmos exatos de maneira eficiente. Desta forma, várias propostas na literatura fazem uso de métodos aproximativos como metaheurísticas (MHs) e algoritmos evolutivos (Hoos e Stuetzle, 2004). Porém, muitos destes trabalhos não trazem análises sobre para quais tipos de problemas o algoritmo ou seus parâmetros são mais apropriados. Dessa forma, frente a um novo problema o pesquisador praticante deve implementar e testar várias combinações de algoritmos e parâmetros, o que pode demandar muito tempo e esforço computacional (Pappa, 2014).

Este trabalho tem como objetivo propor e analisar o uso de Programação Genética (GP, do inglês *Genetic Programming*) (Koza, 1994) para recomendação e análise de parâmetros da MH *Iterated Local Search* (ILS) no problema base *flowshop*. *Flowshop* é um problema de otimização combinatória cujo objetivo é encontrar a melhor sequência de execução de tarefas em uma linha de produção com máquinas em série (Pinedo, 2016). A ILS (Hoos e Stutzle, 2004) é uma MH bastante genérica para gerar algoritmos com diferentes características.

Na literatura é possível encontrar diversas técnicas para ajuste de parâmetros como REVAC (Nannen e Eiben, 2007), ParamILS (Hutter et al, 2009) e *Irace* (Lopez-Ibanez et al, 2016). Muitas destas propostas ajustam parâmetros em um conjunto de instâncias. Embora este trabalho não busque o desenvolvimento de um sistema de meta-aprendizado (Pappa, 2014), ele poderá ser auxiliar uma vez que propõe um algoritmo baseado em GP, denominado GP-*tuner*, para encontrar parâmetros da meteheurística ILS em instâncias individuais do *flowshop*.

MATERIAIS E MÉTODOS

O *flowshop* é caracterizado por um conjunto de M máquinas em série para realizar de N tarefas, de forma que cada tarefa seja processada em cada máquina e elas sejam executadas na mesma ordem. Nesse cenário, um objetivo comum é encontrar o escalonamento com menor tempo total para processar todas as tarefas, conhecido como *makespan* (Pinedo, 2016). MHs são procedimentos de busca para encontrar uma solução suficientemente boa para o problema. Contudo, MHs ainda dependem do ajuste dos seus parâmetros (numéricos ou categóricos) para que atuem de forma satisfatória. A ILS é uma MH de trajetória que, de uma solução inicial, iterativamente busca por melhores soluções na vizinhança e usa operadores de perturbação para evitar ótimos locais (Hoos e Stuetzle, 2004).

A ILS pode ser configurada para utilizar diferentes inicializações e buscas locais. A inicialização pode ser aleatória (RND) ou seguir a heurística NEH (Nawaz, Encore e Ham, 1983) (NEH ou RNEH). A comparação entre soluções pode ser estrita (STRICT) ou não (EQUAL). A busca local pode ser baseada na primeira solução da vizinhança que melhora a solução atual (*first improvement*, FI) ou baseada nas melhores soluções (BEST ou RBEST). Os vizinhos, ou parte deles, são percorridos de maneira ordenada (ORDERED) ou aleatória (RANDOM). Na perturbação, a ILS investigada faz uso do algoritmo IG (Ruiz e Stutzle, 2007) que possui como parâmetro o número de desconstruções. O critério de aceitação possui um parâmetro regularizador de temperatura. É possível notar que os parâmetros da ILS incluem três parâmetros dependentes da busca local: a comparação utilizada, o tipo de vizinhança e o tamanho da vizinhança.

Algoritmos evolutivos são algoritmos iterativos baseados na evolução natural, onde indivíduos mais adaptados sobrevivem e novos indivíduos são gerados através da reprodução. Um tipo de algoritmo evolutivo é a programação genética (GP) onde os indivíduos são usualmente representados como árvores (Engelbrecht, 2007). A vantagem da GP neste contexto é a possibilidade de especificar uma hierarquia de parâmetros, tal qual acontece na ILS.

ALGORITMO PROPOSTO: GP-*TUNER*

O algoritmo proposto, denominado GP-*tuner*, emprega conceitos de programação genética para o ajuste de parâmetros de MHs. No GP-tuner, o indivíduo representa os parâmetros da MH e a população de tamanho N é evoluída por G gerações. A população é inicializada com valores aleatórios e, a cada geração, são gerados 2N novos indivíduos. Para isso, o algoritmo itera por cada um dos indivíduos na população, escolhendo-o como pai para gerar novos filhos. O segundo pai é escolhido por meio de um torneio. Dois filhos são gerados por *crossover* e mutação. No *crossover* há troca de informação genética e na mutação o valor é gerado aleatoriamente. Na seleção, 2N piores indivíduos são eliminados.

O GP-*tuner* respeita a hierarquia dos parâmetros ao aplicar os operadores evolutivos. Quando ocorre o *crossover* de um parâmetro que possui outros parâmetros abaixo dele na hierarquia da MH (como é o caso da busca local da ILS), tanto o parâmetro sofrendo *crossover* quanto seus descendentes são permutados entre os filhos sendo gerados.

RESULTADOS

Nesta seção o algoritmo proposto é testado para ajustar parâmetros da ILS em várias instâncias do problema *flowshop*. O GP-*tuner* é comparado com o estado da arte para ajuste de parâmetros, o *Irace*. Após isso, os valores dos parâmetros ajustados são investigados. As instâncias são geradas de acordo com Taillard (1993) e possuem 20 ou 30 tarefas (J) com 10 ou 20 máquinas (M). Para cada tamanho, são geradas 10 instâncias com tempos de processamento diferentes, totalizando 40 instâncias do *flowshop*.

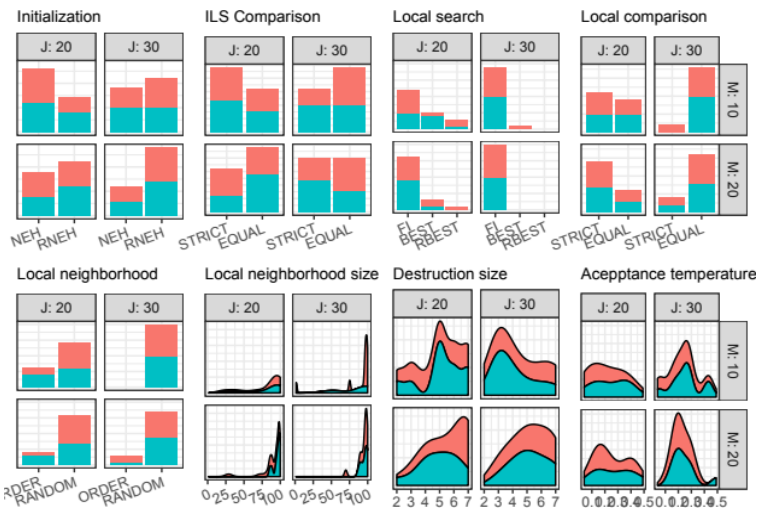
Os algoritmos testados têm como critério de parada um número máximo de 10 mil ou 40 mil avaliações de configuração. Uma configuração é avaliada no problema base pela ILS com um número máximo de iterações (avaliações do problema base) igual a 10 x M x J avaliações da função objetivo (*makespan*). A melhor configuração resolve a instância 30 vezes e a média do *makespan* é utilizada para comparação entre os algoritmos. Além da comparação de médias, o teste estatístico de Kruskal-Wallis com 95% de confiança foi utilizado para determinar se há diferença estatística entre os valores de função objetivo.

Para comparação com a literatura, o GP-*tuner* foi comparado com o algoritmo *Irace*. Testamos o *Irace* com parâmetros padrão em comparação com o GP-*tuner* com população N = 20. A especificação do espaço de parâmetros para o *Irace* é similar ao GP-*tuner*, incluindo a noção de hierarquia dos parâmetros.

O Quadro 1 mostra a comparação entre o GP-*tuner* e o *Irace* com 10 mil e 40 mil avaliações de configuração, para cada tamanho de instâncias utilizadas. Para 10 mil avaliações, o GP-*tuner* obteve melhor média em 28 de 40 instâncias e 13 destas foram estatisticamente melhores. Em 11 instâncias o *Irace* obteve melhor desempenho, das quais 4 foram estatisticamente melhores. Para 40 mil avaliações de configuração, o GP-*tuner* obteve melhor média em 23 instâncias, sendo 13 estatisticamente melhor. Já o *Irace* foi o melhor em 16 instâncias e em 4 destas houve diferença estatística. Em uma instância com 20 tarefas e 20 máquinas todas as configurações encontradas resultaram no mesmo *makespan*.

A Figura 1 mostra as frequências dos parâmetros encontrados pelo GP-tuner para cada tamanho de instância testado. Considerando os parâmetros categóricos, a inicialização NEH é a mais presente em instâncias pequenas com J = 20 e M = 10 e o NEH aleatório é mais recomendado em instâncias maiores com J = 30 e M = 20. A comparação estrita do ILS é um pouco melhor em instâncias pequenas. A busca local FI foi a mais recomendada para instâncias maiores com J = 30. A vizinhança aleatória foi a mais utilizada na maioria dos casos. Nos parâmetros numéricos, o tamanho da vizinhança é próximo de 100%. O número de desconstruções IG é próximo de 5 para instâncias menores e 3 em instâncias com J = 30 e M = 10. A temperatura de aceitação não possui grande influência em instâncias com J = 20 e assume valores próximos de 0.25 quando J = 30.

Figura 1: Frequência de cada valor de parâmetro nas configurações obtidas pelo GP-*tuner*.



CONCLUSÕES

O presente trabalho propõe e analisa um novo método para ajuste de parâmetros, denominado GP-*tuner*. O GP-*tuner* foi utilizado para encontrar parâmetros da ILS no problema de otimização combinatória *flowshop*. Os parâmetros da ILS possuem uma estrutura hierárquica, dessa forma a proposta é baseada em Programação Genética que busca por uma configuração ótima para cada instância. O algoritmo foi comparado com o estado da arte para ajuste de parâmetros, o *Irace*, e apresentou resultados promissores.

Por fim, foram realizadas algumas análises preliminares sobre os resultados obtidos pelo GP-*tuner*. Em trabalhos futuros, espera-se utilizar modelos de aprendizado de máquina para construção de um sistema de meta-aprendizado. Da mesma forma, o GP-*tuner* pode ser testado com mais meta-heurísticas e diferentes problemas de otimização.

REFERÊNCIAS

HOOS, H. H.; STÜTZLE, T. **Stochastic local search: Foundations and applications**. [S.l.]: Elsevier, 2004.

PAPPA, G. L. et al. **Contrasting meta-learning and hyper-heuristic research: The role of evolutionary algorithms**. Genetic Programming and Evolvable Machines, Kluwer Academic Publishers, Hingham, MA, USA, v. 15, n. 1, p. 3–35, 2014. ISSN 1389-2576. Disponível em: <http://dx.doi.org/10.1007/s10710-013-9186-9>

KOZA, J. R. **Genetic programming as a means for programming computers by natural selection**. Statistics and computing, Springer, v. 4, n. 2, p. 87–112, 1994.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 5. ed. [S.l.]: Springer, 2016.

NANNEN, V.; EIBEN, A. E. **Relevance estimation and value calibration of evolutionary algorithm parameters**. In: IJCAI. [S.l.: s.n.], 2007. v. 7, p. 975–980.

HUTTER, F. et al. **Paramils: an automatic algorithm configuration framework**. Journal of Artificial Intelligence Research, v. 36, n. 1, p. 267–306, 2009.

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

NAWAZ, M.; ENSCORE, E. E.; HAM, I. **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**. Omega, v. 11, n. 1, p. 91 – 95, 1983. ISSN 0305- 0483.

RUIZ, R.; STüTZLE, T**. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem**. European Journal of Operational Research, v. 177, n. 3, p. 2033 – 2049, 2007.

ENGELBRECHT, A. P. **Computational Intelligence: An Introduction**. 2nd. ed. [S.l.]: Wiley Publishing, 2007. ISBN 0470035617.

TAILLARD, E. **Benchmarks for basic scheduling problems**. European Journal of Operational Research, v. 64, n. 2, p. 278 – 285, 1993. ISSN 0377-2217.

AGRADECIMENTOS

Agradecemos à Universidade Tecnológica Federal do Paraná por ceder o espaço para as pesquisas e ao CNPq pelo fornecimento da bolsa estudantil.