Projeto de Metaheurísticas

Regras para o trabalho:

- O trabalho pode ser realizado individualmente ou em duplas;
- A entrega será feita pelo *moodle*, na data limite de 24/07/2022;
- O trabalho será apresentado em sala de aula no dia 25/07/2022;
- Todos os arquivos necessários para execução do projeto (incluindo instruções de execução) devem ser incluídos na postagem.

O trabalho consiste no estudo e projeto de metaheurísticas para a solução de problemas de otimização complexos. Nesse processo, vocês deverão: selecionar um problema (pesquisando artigos científicos e selecionando instâncias da literatura); fazer um estudo inicial da paisagem de otimização; implementar um conjunto de metaheurísticas, incluindo estratégias construtivas e por modificação (perturbativas); aplicar uma metodologia de projeto e configuração desses algoritmos; desenvolver um estudo experimental completo, comparando com outras abordagens da literatura; apresentar um relatório com detalhes do desenvolvimento do trabalho e dos resultados obtidos.

Abaixo são apresentados os problemas disponíveis e alguma referência de exemplo. Cada problema poderá ser selecionado por no máximo duas equipes, as quais deverão informar o problema selecionado em fórum específico no *moodle*. Mais detalhes sobre os problemas podem ser discutidos com o professor.

Problema	**Referência(s)**
Programação quadrática binária (UBQP)	de Souza and Ritt [2018b]; Kochenberger et al. [2014]
Empacotamento (bin packing)	Lodi et al. [2004]; de Souza et al. [2022]
Alocação de testes	Duives et al. [2013]; de Souza and Ritt [2018a]
Caixeiro viajante	Applegate et al. [2011]; de Souza et al. [2022]
Diversidade máxima (MDP)	Martí et al. [2013]; Ghosh [1996]

Abaixo são apresentados os algoritmos (metaheurísticas) disponíveis para seleção. Os algoritmos básicos (heurísticas construtivas e buscas locais simples) já serão implementados no estudo inicial, portanto a lista apresenta somente as demais abordagens estudadas em sala de aula.

Modificação de soluções:

- busca local com múltiplos inícios
- busca local randomizada
- busca tabu
- busca local iterada

Construção de soluções:

- algoritmos semi-gulosos: guloso-k e guloso-α
- construção repetida
- GRASP
- iterated greedy (guloso iterado)

O trabalho será dividido nas etapas apresentadas abaixo. Cada etapa deve estar detalhada no relatório, cujo *template* LaTeX está disponível no *moodle*.

Etapa 1: seleção do problema e dos algoritmos

Ao selecionar o problema de otimização desejado, vocês devem buscar ao menos dois artigos científicos que apliquem alguma metaheurística para sua solução, que apresentem um estudo experimental e que usem instâncias disponíveis *online*. Esses artigos servirão de base para o estudo a ser realizado e as abordagens ali propostas serão comparadas com os algoritmos desenvolvidos. Os artigos selecionados deverão ser informados no fórum do trabalho (disponível no *moodle*), juntamente com a seleção do problema. Esses artigos deverão ser discutidos e aprovados pelo professor.

Além disso, deve ser selecionado um subconjunto de pelo menos cinco das metaheurísticas estudadas em sala de aula (ao menos uma de cada categoria). Esses algoritmos serão implementados e comporão o espaço algorítmico usado no projeto do algoritmo final. As metaheurísticas selecionadas também devem ser discutidas e informadas no fórum.

Etapa 2: estudo inicial

No estudo inicial, deverão ser implementadas duas heurísticas simples (construtivas) que gerem soluções para o problema. Essas heurísticas devem ser comparadas com uma caminhada aleatória no espaço de busca e com uma buscas locais simples (que usam estratégias de seleção de primeira melhora e melhor melhora). Deve ser apresentada uma tabela com os resultados dos cinco algoritmos e também da(s) abordagem(ns) da literatura (i.e. dos artigos selecionados). Uma análise dos resultados deve ser apresentada, mostrando o desempenho das diferentes abordagens e quão longe das melhores soluções conhecidas estão estratégias simples. Para o estudo inicial, selecionem 20 instâncias da literatura, com base nos experimentos apresentados em um dos artigos selecionados.

Etapa 3: implementação baseada em componentes

Os algoritmos escolhidos deverão ser implementados, tendo como foco a combinação de diferentes abordagens para a produção de algoritmos híbridos. Por exemplo, o algoritmo GRASP pode usar qualquer heurística construtiva (e.g. guloso-k ou guloso-α), bem como qualquer busca local (e.g. busca tabu ou busca local iterada). Logo, as diferentes abordagens devem ser implementadas de modo que possam ser usadas como procedimentos internos de outros algoritmos.

A instância e todos os parâmetros necessários para a execução devem ser lidos na chamada do algoritmo (i.e. via terminal). Quando não fornecidos valores na chamada, deve-se adotar valores padrão pré-estabelecidos (os parâmetros serão calibrados em uma etapa futura). Como saída, o algoritmo deve imprimir o valor da função objetivo da melhor solução encontrada. Opcionalmente, o algoritmo pode imprimir informações adicionais que podem ser usados na análise experimental, como o número de iterações ou o perfil de desempenho (cada melhor solução encontrada ao longo da execução juntamente com o tempo gasto para encontrá-la).

Teste os diferentes algoritmos e várias combinações de componentes para se certificar de que a implementação está correta.

Etapa 4: estudo de um algoritmo adicional

Além dos algoritmos implementados, cada equipe deve selecionar uma metaheurística para estudar e apresentar em sala de aula (não é necessário implementá-la). A apresentação deve abordar a ideia geral do algoritmo e seus detalhes de funcionamento. Abaixo são apresentadas as opções (a escolha deve ser indicada no *moodle* e não deve haver repetição).

- Recombinação de soluções
  1. PROBE
  2. Busca dispersa (scatter search)
  3. Algoritmos genéticos
  4. GRASP+PR (GRASP com path relinking)
- Swarm intelligence
  1. Otimização por colônia de formigas (ACO)
  2. Colônia de abelhas artificial (ABC)
  3. Otimização por enxame de partículas (PSO)
- Buscas locais adicionais
  1. Têmpera simulada (simulated annealing)
  2. Busca com vizinhança variável (VNS)

Etapa 5: geração de algoritmos e seleção do melhor

Com base nos algoritmos/componentes implementados na etapa 3, cada equipe deverá gerar seis algoritmos híbridos (ou seja, combinando componentes implementados). Exemplos de combinações são:

- Busca local iterada com {primeira melhora, melhor melhora, busca tabu} e ainda com diferentes estratégias de perturbação;
- GRASP com {guloso-k, guloso-α} e ainda com {busca local randomizada, busca tabu};
- Guloso iterado com {guloso-k, guloso-α} e ainda com diferentes estratégias de destruição.

Deverão ser selecionadas cinco instâncias (das 20 usadas no estudo inicial da etapa 2) para uma avaliação prévia desses seis algoritmos. As instâncias devem ser representativas, ou seja, devem abranger as características do conjunto completo (sobretudo a distribuição de tamanhos). O desempenho dos seis algoritmos serão avaliados com base nessas instâncias, de modo a identificar o melhor deles. Abaixo é fornecido um resumo da metodologia do experimento.

1. Para cada instância, executar cada algoritmo com pelo menos 5 replicações, usando o critério de terminação predeterminado (com base no artigo selecionado, com o qual o melhor algoritmo será comparado);
2. Para cada execução, calcular o desvio médio relativo D;
   - Para um problema de maximização, $D = v/o$, onde v é o valor da melhor solução encontrada e o é o valor da solução ótima (ou algum limitante, como o valor da melhor solução conhecida);
   - Caso o artigo selecionado use uma medida diferente, essa poderá ser adotada para fins de comparação.
3. Calcular a média de D para cada par {algoritmo, instância}, bem como a média geral para cada algoritmo;
4. Produzir uma tabela com esses valores (passo anterior), identificando o algoritmo com melhor desempenho.

Etapa 6: avaliação do melhor algoritmo

Uma vez identificado o melhor algoritmo produzido, a mesma metodologia experimental da etapa 5 deve ser repetida, mas usando o conjunto completo de instâncias. Ao final desta etapa, deverá ser produzida uma tabela completa, comparando as seguintes abordagens: a melhor busca local simples da etapa 2; o melhor algoritmo produzido (identificado na etapa 5); o algoritmo apresentado no artigo selecionado (na etapa 1). Discutam os resultados obtidos, as diferenças de desempenho entre os diferentes algoritmos e possíveis melhorias nas abordagens implementadas.

Etapa 7: relatório

Ao final do trabalho, deve ser entregue um relatório simplificado, contendo as informações referentes a cada etapa. O problema e os algoritmos implementados não precisam ser explicados. As estratégias específicas devem estar detalhadas (e.g. heurísticas construtivas, operador de vizinhança, estratégia de perturbação, ...). Devem ser incluídas as referências (artigos usados, instâncias adotadas, ...) e os detalhes experimentais, bem como as tabelas com os resultados. O relatório deve ter no máximo 3 páginas (formato livre).

Referências

David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The traveling salesman problem. In *The Traveling Salesman Problem*. Princeton University Press, 2011.

Marcelo de Souza and Marcus Ritt. An automatically designed recombination heuristic for the test-assignment problem. In *Proceedings of the 2018 Congress on Evolutionary Computation (CEC 2018)*, pages 2411–2418, Piscataway, NJ, 2018a. IEEE Press. doi: 10.1109/CEC.2018.8477801.

Marcelo de Souza and Marcus Ritt. Automatic grammar-based design of heuristic algorithms for unconstrained binary quadratic programming. In Arnaud Liefooghe and Manuel López-Ibáñez, editors, *Proceedings of EvoCOP 2018 – 18th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 10782 of *Lecture Notes in Computer Science*, pages 67–84. Springer, Heidelberg, Germany, 2018b. doi: 10.1007/978-3-319-77449-7_5.

Marcelo de Souza, Marcus Ritt, and Manuel López-Ibáñez. Capping methods for the automatic configuration of optimization algorithms. *Computers & Operations Research*, 139:1–15, 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105615.

Jelle Duives, Andrea Lodi, and Enrico Malaguti. Test-assignment: A quadratic coloring problem. *Journal of Heuristics*, 19(4):549–564, 2013.

Jay B Ghosh. Computational aspects of the maximum diversity problem. *Operations research letters*, 19 (4):175–181, 1996.

Gary A. Kochenberger, Jin-Kao Hao, Fred Glover, Rhyd M. R. Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: A survey. *Journal of Combinatorial Optimization*, 28(1):58–81, 2014. doi: 10.1007/s10878-014-9734-0.

Andrea Lodi, Silvano Martello, and Daniele Vigo. TSpack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, 131(1-4):203–213, 2004. doi: 10.1023/B:ANOR.0000039519.03572.08.

Rafael Martí, Micael Gallego, Abraham Duarte, and Eduardo G Pardo. Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, 19(4):591–615, 2013.