# 1 GRASP

GRASP, *Greedy Randomized Adaptative Search*, é um arcabouço geral pra solução de problemas de otimização. O emprego da meta-heurística GRASP na resolução de um problema consiste em:

1. Construir de uma solução inicial viável usando uma heurística gulosa
2. Buscar soluções melhores em torno da vizinhança da solução contruida.

Estes dois procedimentos são repetidos quantas vezes se julgar adequado. Os metódos e heurísticas utilizados na construção da solução inicial variam de problema pra problema, bem como a definição de vizinhança de uma solução.

# 2 Fase construtiva

A fase construtiva da minha implementação do GRASP para o **FJSP** aloca uma operação em uma máquina de cada vez até que todas as operações tenham sido alocadas. No caso de operações que podem ser delegadas a mais de uma máquina, uma das máquinas é escolhida aleatoriamente; tornando cada iteração do GRASP em uma instância de **JSP** diferente.

Quando se escolhe qual a próxima operação vai ser alocada, essa escolha deve ser limitada as primeiras operações na sequência tecnológica de cada job para evitar quebras de restrição que levariam a soluções não viáveis. Cada operação recebe um valor de acordo uma heurística, as $n$ melhores são colocadas numa lista restritiva, de onde é escolhida, aleatoriamente, a próxima operação alocada.

A heurística utilizada para avaliar as operação é sua ordem na sequencia tecnológica do respectivo *job*, índices menores são melhor avaliados. Dessa forma, a construção tende a alocar as operações inicias dos *jobs* - aquelas das quais o restante das operações do *job* depende - primeiro.

# 3 Busca local

Uma solução está completamente especificada quando é definida uma orientação para todos os arcos disjuntivos do grafo disjuntivo. São consideradas vizinhas de uma solução, todas as soluções que estão a uma inversão de arco disjuntivo de distância.

A busca local implementada aqui consiste em avaliar todos os vizinhos de uma solução e escolher o melhor, melhor aqui significa aquele que tem o menor caminho crítico da origem do grafo disjuntivo até o fim, o melhor vizinho se torna a solução corrente e uma nova busca local é realizada em torno de seus vizinhos. Quando a busca local não encontra nenhuma solução melhor que a solução corrente, o processo para, e uma nova iteração do GRASP é iniciada.

O caminho crítico é o maior caminho simples da origem até o fim do grafo disjuntivo. Para calcular o caminho crítico, primeiro as arestas são ordenadas topologicamente [3] ao mesmo tempo que se garante que o grafo não contém

ciclos [1], grafos disjuntivos com ciclos representam soluções não viáveis, além de não ser possivel calcular maior distância em grafos com ciclos. Tendo a ordem topologica do grafo a maior distância é calculada como em [2].

# References


[1] https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm

[2] https://en.wikipedia.org/wiki/Longest_path_problem

[3] Thomas H. Cormem - Introduction to Algorithms, secção 22.4 Topological sort