Universidade Federal da Paraíba
Centro de Informática
Disciplina: Análise e Projeto de Algoritmos
Professor: Bruno Bruck

Projeto Final

Descrição do problema

Logo após terminar a faculdade, Ana decidiu ser empreendedora e juntou-se à sua amiga Paula para abrir uma startup que presta consultoria na área de Pesquisa Operacional. A empresa foca principalmente em resolver problemas de otimização combinatória para empresas de diversos setores, utilizando tanto algoritmos exatos quanto heurísticos.

Após três anos difíceis e desafiadores, a empresa está passando por uma ótima fase e vem crescendo bastante nos últimos meses com a entrada de clientes importantes. O modelo de negócios deles foca principalmente no licenciamento de soluções web desenvolvidas internamente pela empresa, de forma que o cliente paga um valor inicial bem menor do que o normal, somente para financiar parte do desenvolvimento e customização, e depois pode utilizar livremente a solução pagando uma assinatura anual. Ana atribui muito do sucesso da empresa a esse modelo. Entretanto, isso gera algumas complicações. Em particular, a execução dos algoritmos de otimização propriamente ditos é o que consome a maior parte dos recursos computacionais da empresa. Todas as vezes que um cliente acessa a aplicação web e requisita uma simulação, é gerado um *job* para a execução do respectivo algoritmo. Tal *job* deve ser alocado e executado em um dos servidores da empresa.

Até o momento, todas as aplicações em funcionamento rodam em servidores locais, adquiridos e mantidos pela própria empresa. Porém, durante a última reunião de planejamento estratégico, foi repassado a Ana e Paula que, para acompanhar a demanda, seria necessário um grande investimento para atualizar e escalar a infra-estrutura. Uma alternativa apresentada foi a manter localmente a hospedagem das aplicações, mas terceirar o processamento dos *jobs* em servidores na nuvem. Nesse caso a empresa paga somente pelo tempo de processamento consumido no servidor contratado.

O funcionamento do novo sistema foi planejado da seguinte forma. Quando um cliente submete uma requisição para a execução de uma simulação, um *job* é criado e adicionado a uma fila. Algumas vezes por dia, o sistema da empresa coleta todos os *jobs* presentes na fila e os distribui entre servidores na nuvem. Como a empresa de Ana e Paula é especialista em Otimização, eles decidiram desenvolver um algoritmo para decidir, dado um conjunto de *jobs*, a melhor alocação nos servidores disponíveis. Para compreender melhor o problema e seus requisitos, considere a seguinte definição formal:

> " Seja n o número de *jobs* e m o número de servidores na nuvem disponíveis no portfólio da empresa. Cada *job* $j \in \{1, \ldots, n\}$ requer um tempo de processamento igual a t_{sj} para ser processado no servidor $s \in \{1, \ldots, m\}$ e gera um custo c_{sj}. Além disso, cada servidor s, possui uma capacidade b_s, que especifica a quantidade máxima de tempo que pode ser contratada. Para cada *job* não alocado em um servidor na nuvem, vamos assumir que este será rodado localmente com um custo fixo p. Sendo assim, o objetivo do problema é o de alocar cada *job*

em um dos servidores, sem violar a capacidade dos mesmos, e de forma a minimizar o custo total para a empresa. ”

Exemplo de instância e solução

Para exemplificar o problema, considere uma instância (cenário) onde $n = 6$, $m = 2$ e $p = 1000$. O vetor de capacidades dos servidores é dado por $b = \{220, 350\}$, e as matrizes t e c são como a seguir:

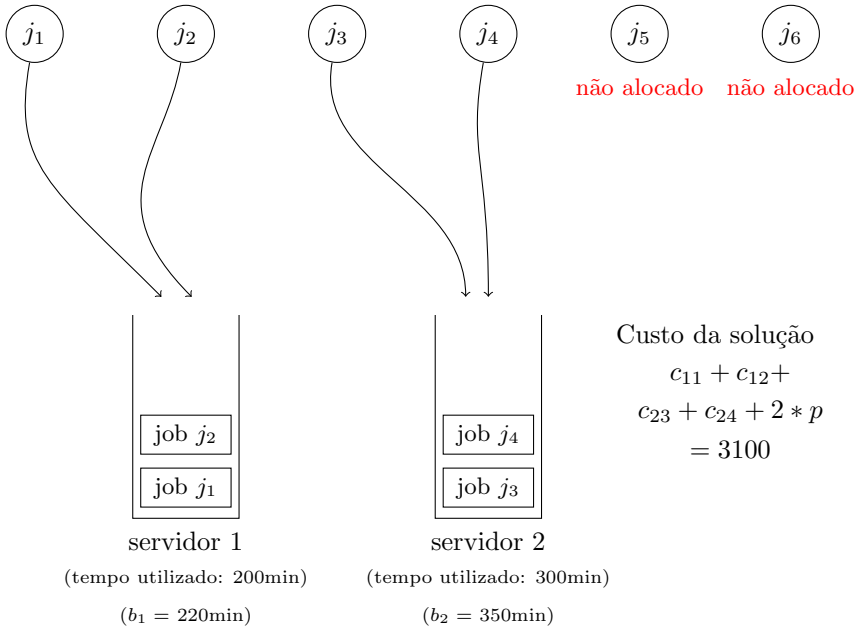
Table 1: Matriz t

	j_1	j_2	j_3	j_4	j_5	j_6
servidor 1	120	80	180	95	35	52
servidor 2	145	70	230	70	40	59

Table 2: Matriz c

	j_1	j_2	j_3	j_4	j_5	j_6
servidor 1	350	50	540	245	145	200
servidor 2	410	80	500	200	100	196

Nesse cenário, uma possível solução seria como a seguir:



Instruções

O projeto deve ser realizado em grupo de **3 integrantes** e vale 10 pontos, relativos à terceira nota da disciplina. Cada grupo deve desenvolver um algoritmo eficiente de busca local (ou meta-heurística) para o problema de otimização descrito acima. O código-fonte deve ser **obrigatoriamente** escrito na linguagem **C/C++**.

Note que o seu programa deve ser capaz de ler um arquivo contendo os dados de uma instância (cenário) do problema e utilizar tais dados como entrada para o algoritmo. O formato de arquivo a ser utilizado é o seguinte:

```
n
m
p

array b

matriz t (servidores x jobs)

matriz c (servidores x jobs)
```

A instância utilizada na seção anterior, por exemplo, poderia ser representada pelo seguinte arquivo:

```
5
2
1000

220 350

120  80 180  95  35
145  70 230  70  40

350  50 540 245 145
410  80 500 200 100
```

Ao final da execução, seu código deve produzir um arquivo de saída, no seguinte formato, contendo a melhor solução encontrada:

```
<valor da solucao>
<custo de alocacao nos servidores>
<custo associado a jobs executados localmente>

<lista de jobs alocados no servidor 1>
<lista de jobs alocados no servidor 2>
...
<lista de jobs alocados no servidor m>
```

Por exemplo, a solução mostrada na seção anterior geraria o seguinte arquivo de saída:

```
3100
1100
2000

1 2
3 4
```

Etapas e prazos

Este projeto contém os seguintes entregáveis:

- Implementação de **ao menos uma heurística de construção**, que nada mais é do que um **algoritmo guloso** para a geração de uma solução viável.
- Implementação de **pelo menos 3 (três) estruturas de vizinhança**.
  Observação: Todas as estruturas devem realizar uma busca exaustiva na vizinhança, o que significa que devem verificar todas as possíveis combinações.
- Implementação do algoritmo de busca local **VND** (Variable Neighborhood Descent).
- Implementação de uma **meta-heurística** (OPCIONAL). Sugestões: GRASP ou ILS.
- Resultados computacionais: **criar uma tabela** que contenha os resultados obtidos pela(s) heurística(s) construtiva(s) e pelo VND, e que compare tais resultados com a solução ótima de cada instância. Essa tabela deverá conter os seguintes dados para cada heurística construtiva e para o VND:
  - Melhor solução encontrada;
  - Média do tempo gasto pelo respectivo algoritmo;
  - GAP para a solução ótima.
  - Caso exista algum fator aleatório no algoritmo, incluir a média do valor da solução, do tempo de execução e da métrica GAP de no **mínimo 10 execuções** para cada instância.

  Observação: Caso decida implementar a meta-heurística, é necessário adicionar uma coluna de resultados para ela na tabela.
- Todas as implementações devem vir acompanhadas de um arquivo ***makefile*** para a compilação. Tal arquivo deve ser preparado de forma a funcionar em sistemas UNIX.
- Criar uma pasta contendo os arquivos de saída gerados durante os testes com cada instância. Favor incluir somente os resultados dos testes finais, com a versão a ser entregue.

IMPORTANTE: O projeto deve ser entregue até às **23:59 do dia 28 de Abril de 2024** (Não haverá adiamento do prazo de entrega). Devem ser enviados via SIGAA, em um arquivo compactado, o **código-fonte** do projeto, o arquivo **makefile**, os arquivos de saída contendo as soluções encontradas, e

um relatório em *pdf* contendo o nome dos integrantes do grupo e a tabela de resultados computacionais. Note que não serão aceitos envios por e-mail ou fora do prazo.

Avaliação

Cada grupo deverá apresentar presencialmente o projeto em data a ser agendada pelo professor. A nota do projeto é individual e leva em consideração diversos critérios, como demonstração de entendimento do código na apresentação, qualidade do código, eficiência dos algoritmos implementados, qualidade dos resultados obtidos, dentre outros. Não apresentar o projeto implica em nota zero.

Dicas

Como calcular o valor da medida GAP: Suponha que desejamos calcular o valor GAP para o resultado da heurística construtiva para uma instância qualquer. Supondo que o valor encontrado pela heurística para essa instância é dado por $valor_{heuristica}$ e o valor ótimo para essa instância é $valor_{otimo}$, o cálculo do GAP é realizado da seguinte forma:

$$gap = \left(\frac{valor_{heuristica} - valor_{otimo}}{valor_{otimo}} \right) \times 100$$

Note que o valor do gap é dado em percentagem (%) e indica a "distância" da solução, no caso, da heurística construtiva para o valor ótimo.

Para calcular o GAP dos resultados obtidos pelo VND basta substituir $valor_{heuristica}$ pelo valor encontrado pelo VND.

Exemplo de tabela de resultados:

		Heurística construtiva			VND		
	ótimo	valor solução	tempo	gap	valor solução	tempo	gap
instancia1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia4	0.0	0.0	0.0	0.0	0.0	0.0	0.0