MC658 – Relatório do Laboratório 2

Marcelo Teixeira ra:122128
Guilherme Sena ra:122924

Nesse laboratório foi proposto a implementação de dois algoritmos de força bruta para um problema NP. Sendo um algoritmo de backtracking e um de branch-and-bound.

Algoritmo Backtracking:

Esse algoritmo foi implementado de forma a ser pouco otimizado, mas relativamente simples, sem cortes na árvore de possibilidades e sem fazer cálculos extras sobre os dados de entrada. Fizemos esse algoritmo da seguinte forma:

```
// funcao recursiva, a primeira chamada passa solucao_temporaria[0...n] = 0
backtracking(solucao, solucao_temporario, total, n){
	if  solucao_factivel(solucao_temporaria) = false then
		return false
	else
		total_temporario = calcula_solucao_temporaria(solucao_temporaria)

	if total_temporario>total then
		total_temporario = total;
		solucao = solucao_temporaria;

	// faz duas chamadas recursivas, uma sem o proximo o usuario, e outra com
	backtracking(solucao, solucao_temporario, total, n-1);
	solucao_temporaria[n]=1;
	backtracking(solucao, solucao_temporario, total, n-1);

	return true;
}
```

Algoritmo Branch-and-Bound:

Nesse algoritmo, devemos podar a árvore de possibilidades, de forma a ter menos caminhos e uma complexidade exponencial amortecida. Nossa estratégia aqui foi o uso do algoritmo guloso, onde ordenamos o vetor de usuarios pelo valor relativo p/w (preço sobre peso), iniciando o vetor de resposta com o primeiro usuario desse vetor, e fazendo um loop com chamadas para o algoritmo passando o próximo usuario da lista na solução.

Para fazer as podas usamos duas condições, primeiro testamos se o ramo é factivel, segundo testamos se o ramo pode ter um valor maior do que o maior já calculado até o momento. O segundo é feito através de um loop que itera em todos os usuarios que ainda podem ser adicionados naquela soluçao, sem considerar seus pesos, pois não é possivel saber quais de fato entrarao na solução.

```
// funcao recursiva, a primeira chamada passa solucao_temporaria[0...n] = 0
bnb(usuarios, solucao, solucao_temporario, total, n){
	if  solucao_factivel(solucao_temporaria) = false then
		return false
	else
		total_temporario = calcula_solucao_temporaria(solucao_temporaria)

	if total_temporario>total then
		total_temporario = total;
		solucao = solucao_temporaria;

	// expande as possibilidades da arvore
	for(i=n->usuarios.length()){
		if existe_melhor_possibilidade(usuarios, solucao_temporaria, n) then
			bnb(usuarios, solucao, solucao_temporario, total, n+1);
	}

	return true;
}
```