

Problema de Designação Generalizada utilizando Busca Tabu Otimização Combinatória

Gustavo A. Martini, Gustavo Macedo, Vinicius G. Drage

¹Curso de ciência da Computação– Universidade Estadual do Oeste do Paraná (UNIOESTE)
Cascavel – PR – Brazil

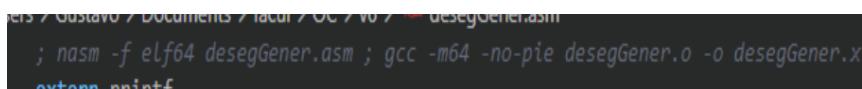
gutamen@live.com, gustavomacedo1366@hotmail.com, vinidrage@gmail.com

Abstract. *This report describes the algorithm implementation process for the generalized assignment problem using tabu search, such as the program's operation and performance.*

Resumo. *Este relatório descreve o processo de implementação do algoritmo para o problema de designação generalizada com a utilização de busca tabu, tal como o funcionamento do programa e seu desempenho.*

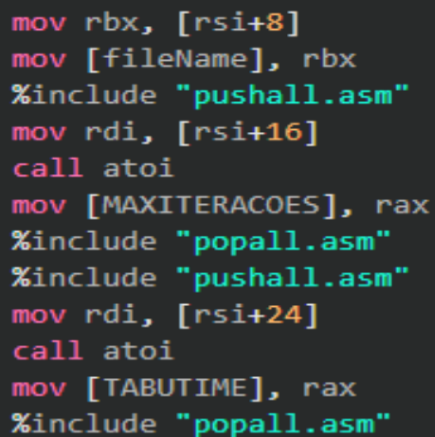
1. Problema e informações gerais

Para a resolução do problema foram criadas funções que realizam etapas de todo o processo, de forma que fique mais organizado, utilizando-se a linguagem Assembly, além de externs em linguagem C. O usuário também pode passar como parâmetro o nome do arquivo a ser lido, e os valores de quantidade de iterações e tempo de tabu.



```
ers / Gustavo / Documents / acad / CC / v0 / - desegGener.asm  
; nasm -f elf64 desegGener.asm ; gcc -m64 -no-pie desegGener.o -o desegGener.x  
extern printf
```

Figure 1. Este comentário contém o comando pronto para criar o executável a partir dos arquivos .asm, necessário linux e nasm instalados.



```
mov rbx, [rsi+8]  
mov [fileName], rbx  
%include "pushall.asm"  
mov rdi, [rsi+16]  
call atoi  
mov [MAXITERACOES], rax  
%include "popall.asm"  
%include "pushall.asm"  
mov rdi, [rsi+24]  
call atoi  
mov [TABUTIME], rax  
%include "popall.asm"
```

Figure 2. Figura mostra a parte do código que capta os parâmetros do programa.

2. Arquivos auxiliares

Para o funcionamento e melhor organização do código, foram criadas funções externas para: Leitura das matrizes e arquivo de entrada, escrita no arquivo de saída e as funções de pop e push da pilha. Dentro da Main também são armazenados uma cópia das matrizes.






 desegGener.asm	27/07/2022 10:46	Arquivo ASM	19 KB
 leitor.asm	22/07/2022 08:10	Arquivo ASM	2 KB
 popall.asm	24/07/2022 15:22	Arquivo ASM	1 KB
 printers.asm	24/07/2022 14:40	Arquivo ASM	2 KB
 pushall.asm	24/07/2022 15:22	Arquivo ASM	1 KB

Figure 3. O programa em sua maior parte se encontra no primeiro arquivo.

3. Cálculo do Arrependimento

Um laço de repetição permite que o código percorra a matriz com os valores, e então seleciona os dois menores consumos de carga horária e faz o cálculo do arrependimento para cada tarefa. Logo em seguida é alocada a lista Tabu que será usada posteriormente. Então os valores dos arrependimentos são passados para um vetor.

```
forRunTarefa:

    mov r15, [ForJ]
    mov edx, [r13+r15*8]
    mov [Memoria1], edx

    movsx r8, dword[quantTarf]
    add r8, r15
    mov ebx, [r13+r8*8]
    mov [Memoria2], ebx
    mov qword[ForI], 2
```

Figure 4. É necessário que existam pelo menos dois programadores para que o programa funcione.

4. Distribuição das tarefas

Em seguida, dentro do vetor com os valores de arrependimento, é a tarefa com o maior arrependimento, enquanto houverem tarefas, a cada tarefa escolhida o arrependimento é recalculado e para que não estoure a carga horária do programador. Isso para o primeiro caso possível, após o primeiro pode ser que um programador estoure a carga horária pagando +2 para cada hora que ultrapassa.

```
parada1:

    mov r15, [ForJ]
    mov r14, [Arrependimento]
    mov eax, [r14+r15*8]
    cmp eax, -1
    je fora
    mov r11d, dword[Memoria2]
    cmp r11d, -1
    je sobrou1
    sub r11d, dword[Memoria1]
    mov [r14+r15*8], r11d
    jmp fora
sobrou1:
    mov r11d, dword[Memoria1]
    mov [r14+r15*8], r11d
fora:
    mov r9d, dword[quantTarf]
    dec r9d
    cmp dword[ForJ], r9d
    je corrigetopofor
    inc dword[ForJ]
    jmp forRunTarefa1
```

Figure 5. Cada seleção move o arrependimento selecionado para -1, assim o programa sabe qual não utilizar novamente.

5. Iterações

Durante as iterações, que tem sua quantidade definida na chamada do programa, são chamadas as seguintes funções: `fprintf`(responsável por escrever no arquivo as informações sobre a iteração atual), `CalculoDisposicao`(responsável por calcular a disposição de troca da tarefa entre os operadores, utilizando o custo como parâmetro), `printVetor`(escreve um vetor no arquivo, no caso das iterações a lista tabu), `TrocaTarefa`(altera a tarefa se baseando no cálculo de disposição), `calculodemehlor`(verifica se a solução, após a troca, é melhor que a anterior e se a mesma é possível).

```
mov ecx, [melhorcusto]
mov rdi, [fileHandle2]
lea rsi, [printIteracao]
mov edx, dword[Iteracao]
xor rax, rax

call fprintf
```

Figure 6. Escreve no arquivo o melhor custo e qual iteração está o programa.

```

;void calculaDisposicao(int *matrizCusto[rdi], int *matrizTrabalho[rsi], int *matriDisposicao[rdx],
; int *matrizCH[rcx], int quanttarefa[r8],int quantprog[r9[r12]], int *tabtabu[rbp+24], int *cargahorialivre[rbp+16])
    mov rdi,[MatrizCusto]
    mov rsi,[Tabcontas]
    mov rdx,[MatrixMut]
    mov rcx,[MatrizCH]
    movsx r8,dword[quantTarf]
    movsx r9,dword[quantProg]
    mov rax,[TabTabu]
    push rax
    mov rax,[CopiaCH]
    push rax

    call CalculoDisposicao
    add rsp, 16
    and qword[rsi-8],0
    and qword[rsi-16],0

```

Figure 7. Cálculo de disposição para cada troca.

```

#include "pushall.asm"
;void[rax] printVetor(int *VetorparaPrint[rdi], int TamanhoLinha[rsi],int *arquivo[rdx])
    mov rdi, [TabTabu]
    movsx rsi, dword[quantTarf]
    mov rdx, [fileHandle2]
    call printVetor
#include "popall.asm"

```

Figure 8. Escreve a lista tabu atual no arquivo.

```

;void[rax] TrocaTarefa(int *matrizTrabalho[rdi], int *matrizdisposicao[rsi],
;int *Listatabu[rdx], int TamanhoMatriz[rcx], int quantTarf[r8], int QualTabu[r9])
    mov rdi, [Tabcontas]
    mov rsi, [MatrixMut]
    mov rdx, [TabTabu]
    movsx rcx, dword[TamanhoMatriz]
    movsx r8, dword[quantTarf]
    mov r9, [TABUTIME]
    mov rax, [MatrizCH]
    push rax
    mov rax, [CopiaCH]
    push rax
    call TrocaTarefa
    add rsp, 16
    and qword[rsp-8], 0
    and qword[rsp-16], 0

```

Figure 9. Troca a tarefa com a maior disposição, que não esteja na lista tabu.

```

;void[rax] calculodemelhor(int *matriztrabalho[rsi], int *melhorcusto,
;int *matrizcusto[rdx], int tamanhomatriz[rcx], int *cargaH[r8], int quantProg[r9])
    mov rdi, [Tabcontas]
    lea rsi, [melhorcusto]
    mov rdx, [MatrizCusto]
    mov rcx, [TamanhoMatriz]
    mov r8, [CopiaCH]
    movsx r9, dword[quantProg]
    call calculodemelhor

;print do custo na iteracao
    xor rax, rax
    mov rdi, [fileHandle2]
    lea rsi, [printatual]
    mov rdx, [custoatual]
    call fprintf

```

Figure 10. Verifica se a solução é possível e maior, caso sim troca a melhor solução para a do estado atual.

6. Tabela Tabu

Ao Trocar de tarefa, o tabu é acionado com o valor que o usuário definiu na chamada do programa(Máximo número de tarefas -1), além disso, durante a troca de tarefa, ocorre o decremento do tempo do tabu para cada tarefa.

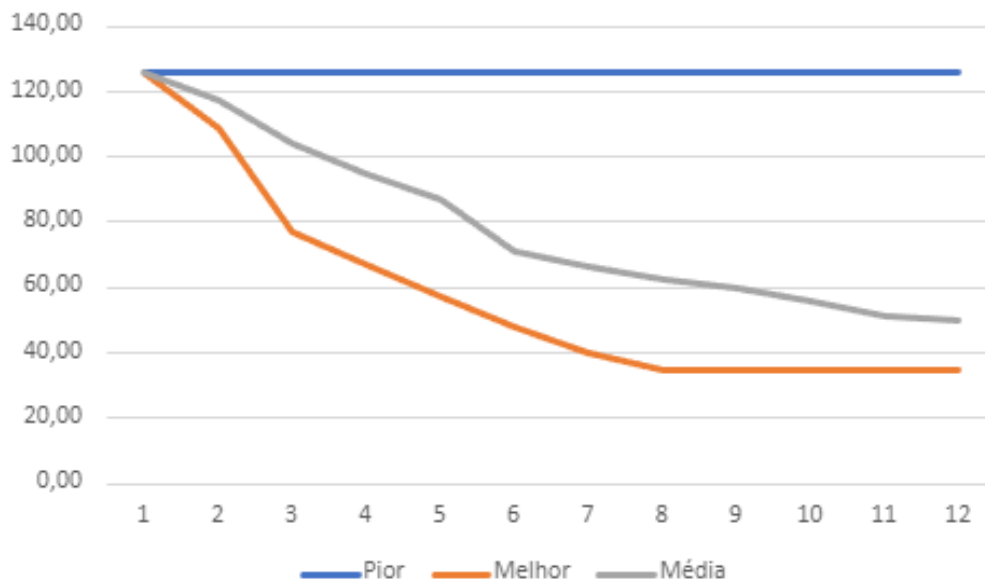


Figure 11. A figura mostra o gráfico da evolução do pior, melhor e médio custo durante 12 iterações, com um tempo de Tabu de 4 iterações, utilizando o arquivo PDG4.txt.