Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Ciência da Computação

PROJETO E ANÁLISE DE ALGORITMOS

Segundo Trabalho Prático - disponível em: http://www.dcc.ufmg.br/~menoti/paa/tp2/

David Menoti Gomes

Professor - Nivio Ziviani

Belo Horizonte
4 de maio de 2004

Sumário

1 Introdução

Neste trabalho é explorado o problema da mochila, (*knapsack problem*). Esse problema é explorado por vários autores na literatura [5],[1],[2],[4]. Existem algumas variações do problema, mas a que se refere ao trabalho em questão é conhecida como 0/1 *knapsack problem*. Então, teve-se o cuidado de selecionar a literatura em relação as abordagens de solução desse trabalho. Ressaltando-se que em [4] encontram-se algoritmos de fácil implementação e de curto tamanho, embora esses não respeitem a definição do problema da mochila que aqui se apresentará. A melhor referência para o problema aqui tratado é [1] para as abordagens *Backtracking* e *Dynamic Programming*. Já para a abordagem *Greedy*, em [2] é apresentado uma solução interessante.

Alguns exemplos práticos para ilustrar o problema da mochila: Um escoteiro-mirim se prepara para um acampamento e neste momento, em fase final de preparativos, ele vai colocar os embrulhos dentro de sua mochila. Mal ele começou, já notou que deverá deixar alguns itens para trás, pois como ele vai caminhar muito, o peso de sua mochila não deverá execeder um limite de M quilos. Para auxiliar no processo de decidir quais itens levar, ele assinalou a cada um deles, um valor que representa a sua utilidade. O problema da mochila [1],[5],[2] é então o problema de decidir, em função das suas utilidades, quais itens levar de modo a não sobrecarregar a mochila.

Outra situação ilustra o mesmo problema. Uma pessoa recebe um prêmio que permite escolher quaisquer dos n itens em uma loja. O i-ésimo item tem valor p_i reais e pesa w_i quilos ($p_i, w_i \in N$). A restrição do prêmio é que a pessoa tem que levar consigo uma mochila que pode carregar até M quilos e $M << \sum_{i=1}^{n} w_i$. Determine o conjunto de itens que a pessoa deve colocar na mochila de forma que ela consiga carregá-los todos dentro da mochila e o valor total dos itens seja maximizado.

Enunciando este problema em termos mais formais: É dado um conjunto C_n de n itens, representado por $C_n = \{1, 2, \ldots, n\}$, cada item $i \in C_n$ tem um peso w_i e utilidade p_i ($w_i > 0, p_i > 0$). Deseja-se determinar um subconjunto S dos itens tal que a soma dos pesos dos elementos de S seja menor ou igual a capacidade da mochila M e que a utilidade total dos elementos de S seja a maior possível. Matematicamente:

$$max \sum_{i \in S} p_i$$

$$\text{sujeito a } \sum_{i \in S} w_i \leq M$$

$$S \subseteq C_n$$

Modificando a representação do problema, mas mantendo-o da mesma forma, pode-se trocar o subconjunto solução S por um conjunto solução X de n elementos, onde o i-ésimo elemento $x_i \in X$ é 1 caso o elemento $i \in C_n$ seja parte da solução e 0 em caso contrário. Pode-se então definir uma nova formulação matemática para o problema.

$$max \sum_{i=1}^{n} x_i \times p_i$$

$$\text{sujeito a } \sum_{i=1}^{n} x_i \times w_i \leq M \tag{1}$$
$$x_i = 0 \text{ ou } 1, 1 \leq i \leq n$$

O objetivo dessa nova formulação é compatibilizar a formulação matemática do problema com a forma exposta por [1].

Nesse trabalho são apresentadas algumas abordagens para resolver o problema da mochila. Duas implementações da abordagem de Tentativa e Erro (*Backtracking*) que apresentam solução ótima, mas com complexidade de tempo exponencial, sendo portanto um algoritmo não eficiente para o problema. Mais duas implementações da abordagem Programação Dinâmica (*Dynamic Programming*) que também apresentam solução ótima, e também tem algoritmo com complexidade de tempo exponencial no pior caso, e também complexidade exponencial para o espaço. E, por último, três implementações de abordagem Gulosa (*Greedy*) apresentam soluções aproximadas para o problema da mochila com complexidade de tempo e espaço linear.

O trabalho está organizado da seguinte forma: Na Seção 2 é apresentada uma prova de que o problema da mochila é *NP*-completo; Na Seção 3 são apresentadas as estruturas de dados utilizadas no trabalho; Os conjuntos de instânicas utilizadas, e a forma como são geradas, são apresentas na Seção 4; Na Seção 5 são apresentadas abordagens para solucionar o problema da mochila. Na Seção 6 são apresentados experimentos e análises desses. Um sistema desenvolvido para gerenciar todas essas funcionalidades é apresentado na Seção 7. As conclusões referentes ao trabalho são mostradas na Seção 8. E, finalmente, as tabelas dos experimentos e os códigos referentes as implementações são anexados na Seção 9.

2 Prova: O problema da Mochila é *NP*-completo.

A prova de que um problema é *NP*-completo é realizada da seguinte forma, em dois passos: Primeiro, deve-se mostrar que o algoritmo está na classe *NP*. Para isso deve-se apresentar um algoritmo não-deterministico que resolve em tempo polinomial o problema ou apresentar um algoritmo deterministico que verifique a solução em tempo polinomial. Segundo, deve-se mostrar uma redução polinomial de um problema que está na classe *NP*-Completo para o problema em questão.

Primeiro (mostrar que o problema pertence a NP), seja o Algoritmo determinista apresentado no Algorítmo 1 que verifica uma solução X, onde $\{x_i \in X | x_i = \{0,1\}\}$, em tempo polinomial, para o conjunto C_n, de n objetos, onde cada objeto i tem peso w_i e utilizade p_i, a capacidade da mochila é M.

Algorítmo 1 Algoritmo determinístico para verificar a solução X em tempo polinomial.

```
procedure VerifySolution(n, M, P, W, X, ppsol)
{n is the number of objects C_n}
{M is knapsack capacity}
{P(1..n) is the set of p_i's such that 1 ≤ i ≤ n}
{W(1..n) is the set of w_i's such that 1 ≤ i ≤ n}
{X(1..n) is the set of x_i's such that 1 ≤ i ≤ n that say if i belongs or don't to
solution set S}
begin
pp ← 0;
ww ← 0;
for i = 1 to n do
   pp ← pp + x_i × p_i;
   ww ← ww + x_i × w_i;
end for
if ww ≤ M then
   return success {solution verified!}
else
   return fail
end if
end
```

Segundo (redução polinomial a partir de outro problema *NP*-Completo), apresenta-se uma redução polinomial de um problema NP-completo conhecido para o problema $0/1$ *Knapsack*. Ou seja, o problema **Cobertura Tripla Exata** (*3-EXACT-COVER*) [3], em *NP*-completo, apresenta uma transformação polinomial para o problema 0-1 *Knapsack*.

Prova 1 *Ele está obviamente em NP; pode-se mostrar que o problema de cobertura tripla exata apresenta uma transfomação polinomial para o problema da mochila (0-1* knapsack*). Iniciando com qualquer família F de n conjuntos de cardinalidade três, deve-se construir inteiros $c_1, c_2, \ldots, c_n$ e K de forma que exista um subconjunto de c_i's, cujo somatório seja K se e somente se existe uma subfamília de F cobrindo exatamente o universo $S = \{u_1, \ldots, u_{3m}\}$.*

Pode-se imaginar todos os conjuntos em F como vetores de bits de comprimento $3m$; por exemplo, $\{u_1, u_2, u_6\}$ e $\{u_3, u_4, u_5\}$ tornando-se 110001 *e* 001110*, respectivamente. Agora, interpreta-se esses vetores de bits como inteiros escritos no sistema de base* $(n+1)$*. Em outras palavras, o inteiro corresponde a um conjunto S_j é*

$$c_j = \sum_{u_i \in S_j} (n+1)^{i-1}.$$

Seja, também, K o inteiro correspondente a $\underbrace{11\ldots1}_{3m}$*:*

$$K = \sum_{j=0}^{3m-1} (n+1)^j.$$

Agora, declara-se que existe uma subfamília de F cobrindo exatamente $\{u_1, \ldots, u_{3m}\}$ se, e somente se, existe um subconjunto de c_j's cujo somatório seja K.

Se, *Suponha que existe um conjunto $S \subseteq \{1, 2, \ldots, n\}$ tal que $\sum_{j \in S} c_j = K$. Realizando esse somatório na base aritmética $(n+1)$, observa-se que somente os dígitos* 0 *e* 1 *aparecem no somatório, e, além disso, o número de somas realizadas é menos que $n+1$, a base. Portanto não há "*carry*"(vai um) nesse processo de adição. Consequentemente, se $\sum_{j \in S} c_j = K$, isso significa que existem exatamente um* 1 *em cada uma das $3m$ posições, ou equivalentemente, a subfamília $C = \{S_j : j \in S\}$ cobre exatamente $\{u_1, u_2, \ldots, u_{3m}\}$.*

E somente se, *Começando de uma cobertura total C de $\{u_1, u_2, \ldots, u_{3m}\}$ observa-se imediatamente que $\sum_{S_j \in C} c_j = K$. Isto conclui a prova.*

Dessa forma, mostra-se que o problema da mochila (0/1 *Knapsack*) pertence a classe de problemas *NP*-completo.

3 Estruturas de Dados

Antes que as abordagens sejam apresentadas, faz-se necessário introduzir as estruturas de dados utilizadas no trabalho. Apesar das abordagens ainda não terem sido apresentadas formalmente, serão discutidas, de forma breve, as estruturas utilizadas pelas abordagens de Tentativa e Erro (*Backtracking*, Programação Dinâmica *Dynamic Programming* e Gulosa (*Greedy*).

Algumas estruturas de dados muito utilizadas nesse trabalho são apresentadas no Algoritmo 1. A estrutura *Object* é utilizada para armazenar cada objeto ou item do conjunto C_n. Esse objeto (i) é composto pela utilidade P (p_i) e pelo peso W (w_i) e um campo chave *Key* que é utilizado para ordenação da coleção de objetos C_n, representado pela estrutura *Vector* (coleção de *Object*'s). Aqui, adianta-se que as abordagens *Backtracking* e Programação Dinâmica e *Greedy* utilizarão esse campo *Key* como sendo a razão entre a utilidade e o peso, como o peso e a utilidade. Maiores detalhes sobre o campo *Key* serão abordados na Seção 5.

```
/* Object structure */
typedef double KeyKind;
typedef struct Object
{
  KeyKind Key;
  /* Another components - satellite data */
  int P;    /* profit of object */
  int W;    /* weight of object */
} Object;

/* Used kind in algorithm implementation */
typedef int Index;
typedef Object* Vector;
```

Programa 1: Estruturas de dados básicas usadas no trabalho.

A maior estrutura de dados utilizada neste trabalho é o objeto (ou classe) *CKnapSack* apresentada no Programa 2 e contida no arquivo knapsack.h, anexado ao trabalho na Seção 9. Nessa estrutura estão englobados todos os dados (primitivas), inclusive as estruturas apresentadas no Programa 1, e funções (métodos) que implementam as abordagens. Especial atenção deve ser dada às primitivas que manipulam conjunto de dados (*Vector*'s). Entre elas, *m_pData* é um vetor que contém a utilidade (p_i) e o peso (w_i) de cada objeto (*Object*'s). Um vetor de vetores (*Vector*'s - ponteiro de ponteiro) *m_ppData* é utilizado para a alocação dinâmica de memória para as produções S^i da abordagem de Programação Dinâmica. Essa estrutura, a mais elaborada do trabalho, permite que essa abordagem resolva problemas com muitos objetos na mochila. Um objeto *m_oStackTime* da classe *CStackTime* foi utilizado para o gerenciamento do tempo de execução de cada solução. Essa classe é anexada ao trabalho na seção 9.

Elementos de segunda importância são ponteiros de inteiros que se comportam como vetores de inteiros. *m_piX* é um vetor de inteiro que contém a solução encontrada por uma abordagem executada. *m_piY* é um vetor de inteiro que contém a solução parcial encontrada pela abordagem *Backtracking*. *m_piSize* e *m_piSizeM* são vetores que controlam o tamanho de cada produção S^i na abordagem de Programação Dinâmica. *m_piPLeft* e *m_piPL* são vetores utilizados para otimização em uma das duas abordagens de Programação Dinâmica implementadas que possibilitaram um significativo aumento de desempenho tanto em tempo quanto em espaço dessa abordagem otimizada em relação a não otimizada. Isto será mostrado pelos resultados dos experimentos apresentados na Seção 6.

Além dessas já citadas, as primitivas (variáveis) restantes relacionam-se com informações diretas ao problema. O tamanho da mochila M é expressado por m_iM, o número de objetos n por m_in. m_ifw e m_ifp armazenam o peso total da solução encontrada e a utilidade máxima encontrada pelo algoritmo executado, respectivamente. O tempo de estouro por $m\_iOverflowTime$, ou seja, o maior tempo que uma instância pode ficar sendo resolvido. Neste trabalho, esse valor foi ajustado para 5. m_iTime e m_iSpace armazenam o tempo e o espaço da última abordagem executada.

```
class CKnapSack
{
private:
  int m_iM;   /* knapsack capacity */
  int m_in;   /* number of objects */
  int m_ifw;  /* weight of solution */
  int m_ifp;  /* profit of solution */

  int m_iOverflowTime;

  CStackTime m_oStackTime;
  int m_iTime;
  int m_iSpace;

  int m_bOptimal;
  int m_bSolution;
  int m_bOverflow;

  Vector m_pData;   /* Vector containing P and W*/
  int *m_piX;       /* Solution */

  int *m_piY;       /* Backtracking, partial solution */

  int *m_piPLeft;   /* dynamic programming approach */
  int *m_piPL;      /* dynamic programming approach */

  Vector *m_ppData; /* dynamic programming approach */

  int *m_piSize;    /* dynamic programming approach */
  int *m_piSizeM;   /* dynamic programming approach */

public:
  CKnapSack();
  CKnapSack(int _M,int _n,Vector _Data);
  ~CKnapSack();

  int Getn            ();
  int GetM            ();
  int Getfw           ();
  int Getfp           ();
  int GetDataP        (int i);
  int GetDataW        (int i);
  int GetDataSolution (int i);
  int GetTime         () {return m_iTime;}
```

```
  int GetSpace            () {return m_iSpace;}

  void Setn               (int);
  void SetPM              (double);
  void SetData            (Vector _Data);
  void SetDataSolution    (int i,int data);

  int IsOptimal           () {return m_bOptimal;}
  int IsSolution          () {return m_bSolution;}
  int IsOverflow          () {return m_bOverflow;}

  void ShowData                     (FILE *pF=stdout);
  void ShowDatanM                   (FILE *pF=stdout);
  void ShowDataSolution             (FILE *pF=stdout);
  void ShowDataBacktracking         (FILE *pF=stdout);
  void ShowDataDynamicProgramming   (FILE *pF=stdout);

  void PrintState                   (FILE* pF=stdout);

private:
  void StartProcess                   ();
  void StartTime                      ();
  void FinishTime                     ();

  void InitData                       ();
  void InitDataSolution               ();
  void InitDataBacktracking           ();
  void InitDataDynamicProgramming     ();

  void CreateData                     ();
  void CreateDataSolution             ();
  void CreateDataBacktracking         ();
  void CreateDataDynamicProgramming   ();

  void ClearData                      ();
  void ClearDataSolution              ();
  void ClearDataBacktracking          ();
  void ClearDataDynamicProgramming    ();

  void VerifyResult                   ();

  /* Backtracking */
  double Bound1(double p,double w,int k);
  double Bound2(double p,double w,int k,double& pp,double& ww,int& i);

  /* Dynamic Programming */
  void CreateVectorPW(int n);
  void AddPW(int n,int _P,int _W);
  void TraceBack();

  /* Key generator */
  void GenerateKeyByRatio  (Vector pData,int n);
  void GenerateKeyByRatio  (void);
  void GenerateKeyByProfit (void);
  void GenerateKeyByWeight (void);

  /* files */
public:
```

```
    void FileSaveInstance(char* sFileName);
    int  FileLoadInstance(char* sFileName);
    void FileSaveSolution(char* sFileName);
    int  FileLoadSolution(char* sFileName);

    /* Generator */
    void GenerateRandom(int,int);

    /* Approaches */
    /* Backtracking */
    void BKnapSack1();
    void BKnapSack2();

    /* Dynamic Programming */
    void DKnapSack1();
    void DKnapSack2();

    /* Greedy */
    void GKnapSack(int opt);
};
```

Programa 2: A classe *CKnapSack*.

Três variáveis, *m_ bOptimal*, *m_ bSolution*, *m_ bOverflow*, de controle de estado foram criadas. Essas variáveis permitem que seja determinado o estado da solução de uma instância executada pelo sistema.

Uma série de métodos foram implementados. Pode-se apresentar como métodos de interfaceamento as seguintes funções: *Getn*() - obtém o número de objetos na instância; *GetM* - obtém o tamanho da mochila; *Getfw*() - obtém o peso computado pela solução; *Getfp*() - obtém a utilidade computada pela solução; *GetTime*() e *GetSpace*() - retornam o tempo gasto e o espaço utilizado pela última abordagem executada; *GetDataP*(i) - obtém a utilidade, p_i, do i-ésimo objeto da instância; *GetDataW*(i) - obtém o peso, w_i, do i-ésimo objeto da instância; *GetDataSolution*(i) - retorna o estado do objeto i no problema, ou seja, se o objeto i está, (*verdadeiro* - 1) ou não (*falso* - 0) na mochila. *Setn*(n) - ajusta à n o tamanho de objetos no conjunto C_n; *SetPM*(PM) - ajusta o tamanho da mochila de forma que $M = PM\% \times \sum_{i=1}^{n} w_i$; *SetData*(*Vector*) - ajusta o vetor *Vector* aos conjuntos de utilidade e de peso; *SetDataSolution*(*i*,*data*) - ajusta o i-ésimo objeto do conjunto C_n como pertencente (*verdadeiro* - 1) ou não (*falso* - 0) à mochila; *IsOptimal*(), *IsSolution*(), *IsOverflow*() informa o estado - se é ótima, se é solução, se houve estouro, respectivamente - da solução obtida pela última abordagem executada; *ShowData*(), *ShowDatanM*(), *ShowDataSolution*(), *ShowDataBacktracking*(), *ShowDataDynamicProgramming*() - apresentam informações relacionadas à instância corrente em um dispositivo de saída; *PrintState* apresenta os valores das três váriáveis de estado, *m_ bOptimal*, *m_ bSolution*, *m_ bOverflow* em seqüência.

Existem alguns métodos privados que são de uso dos métodos publicos, e que não são relevantes serem citados aqui, um estudo mais minucioso do código, apresentado nos Anexos, na Seção 9, pode revelar suas funcionalidades.

Foram criados alguns métodos para leitura e gravação de instâncias (*FileSaveInstance*(*sName*) *FileLoadInstance*(*sName*)) e de soluções (*FileSaveSolution*(*sName*) *FileSaveSolution*(*sName*)) para os dados que estão contidos no objeto. O padrão dos arquivos de instância foi definido da seguinte forma: a primeira linha contém o número de objetos, n, e o tamanho da mochila,

M, respectivamente, da segunda linha em diante, são dispostos o peso, w_i, e a utilidade, p_i, de cada elemento com $1 \leq i \leq n$, da seguinte forma abaixo:

```
n M
w1 p1
w2 p2
w3 p3
.
.
.
wn pn
```

O padrão para os arquivos de soluções foi definido da seguinte forma: a partir da primeira linha, são colocados o peso, w_i, e a utilidade, p_i, cujo x_i seja igual a 1. Ou seja, no arquivo solução, são colocados somente os pares de peso e utilidade daqueles objetos que fazem parte da solução. Os pares para serem colocados no arquivo de saída são ordenados segundo a relação entre utilidade e peso. Na Figura 1 pode ser visto um exemplo de um arquivo de instância, à esquerda, e uma solução correspondente, a direita.

```
4 6
1 2
2 3
4 7
5 4
```

```
4 7
2 3
```

Figura 1: Exemplos de padrões de arquivos: A esquerda, exemplo de arquivo de instância; A direita exemplo de arquivo de solução.

O método *GeneratorRandom*(MAX,*dt*) permite gerar uma instância aleatória com valores inteiros de pesos e utilidades, variando de 1 a MAX e segundo um dos 4 conjuntos de instâncias sugeridos por Horowitz em [1], página 358. O número de objetos, n pode ser ajutado através do método *Setn*(n). Maiores detalhes sobre os conjuntos de instâncias são fornecidos na próxima seção, Seção 4.

Os métodos restantes são os destinados a executarem as abordagens propostas, ou seja:

- Tentativa e Erro - *Backtracking*
  - *void BKnapSack1*();
  - *void BKnapSack2*();
- Programação Dinâmica - *Dynamic Programming*
  - *void DKnapSack1*();
  - *void DKnapSack2*();
- Gulosa - *Greedy*

- *void GKnapSack*(1);
- *void GKnapSack*(2);
- *void GKnapSack*(3);

Para a abordagem Tentativa e Erro foram implementados dois algoritmos, para a abordagem Programação Dinâmica mais dois algoritmos e para a abordagem Gulosa implementou-se somente um método que comporta três diferentes implementações.

Agora, findando-se a apresentação dos tipos de dados, apresenta-se as instâncias utilizadas neste trabalho.

4 Instâncias

O conjunto de instânicas utilizados neste trabalho segue a sugestão dada por Horowitz em [1], página 358. Nessa sugestão encontram-se quatro tipos de conjunto de dados usando números aleatórios. O primeiro conjunto e o segundo conjunto diferem apenas na faixa de variação de números naturais para o peso e para a utilidade. O primeiro fica na faixa de 1 até 1000, enquanto que o segundo fica na faixa de 1 até 100. Já o terceiro e quarto conjunto de instâncias são gerados através de números naturais aleatórios na faixa de 1 até 100 para o peso e a utilidade é definida como sendo o peso mais dez unidades para o terceiro dataset e o peso mais 10% do peso para o quarto conjunto de instâncias. Formalmente pode-se definir os quatro conjuntos de instâncias da seguinte forma:

(1) pesos e utilidades aleatórios na faixa de [1,1000]

(2) pesos e utilidades aleatórios na faixa de [1,100]

(3) pesos aleatórios na faixa de [1,100] e $p_i = w_i + 10$

(4) pesos aleatórios na faixa de [1,100] e $p_i = 1.1 \times w_i$

Seja a implementação da função que gera um número aleatório entre 0 e 1, segundo uma distribuição uniforme, mostrada no Programa 3.

```
double uniformrand0to1()
{
  return (double)(0xdFFFFFFF&mrand48())/0xdFFFFFFF;
}
```

Programa 3: Função que gera números aleatórios entre 0 e 1.

A função *mrand48*() retorna valores de 2^{31} até -2^{31}. O valor de $0xdFFFFFFF$ corresponde ao maior inteiro com sinal, ou seja 2^{31}, pois o bit mais significativo de um inteiro, de 32 bits, em C está relacionado com o sinal do número. Um E lógico (&) com o valor retornado pela função *mrand48*() torna o número positivo e uma divisão em ponto flutuante com o mesmo 2^{31} faz com que o valor calculado por esta função esteja entre 0 e 1. O retorno dessa função é multiplicado pelo maior valor possível da faixa, MAX, menos uma unidade, $1000 - 1$ para o tipo de conjunto 1 e $100 - 1$ para os tipos de conjunto restantes. Esse resultado da multiplicação deve estar contido no intervalo real de 0 até $MAX - 1$. Então, procede-se a soma de uma unidade a esse resultado, gerando a faixa real de 1 até MAX. Com o objetivo de tornar o valor inteiro e arredondado - o mais próximo inteiro, soma-se ao resultado anterior o valor 0.5 e efetua-se uma truncagem no valor real, transformando-o em inteiro. Toda essa explicação pode ser resumida em:

$$p[i] = (int)(uniformrand0to1() * (MAX - 1) + 1 + 0.5)$$

Vale lembrar que todos os conjuntos construídos possuem somente valores inteiros e positivos tanto para o peso como para a utilidade. Essa sugestão de Horowitz versa somente sobre o valor do peso e da utilidade. Portanto, deve-se estabelecer alguns critérios para variar o número de objetos do conjunto, n, e o tamanho percentual da mochila, PM, relacionado a

$$M = PM \times \sum_{i=1}^{n} w_i.$$

onde M é definido como o tamanho da mochila e é arredondado para o inteiro mais próximo.

Essa variação gera vários conjuntos de instâncias, estes por sua vez, permitem testar em um espectro maior a aplicação das abordagens e dar um suporte estatístico para os experimentos.

Então, utilizou-se vários valores para n - número de objetos, e vários valores percentuais para o tamanho da mochila, PM, em relação a soma total dos pesos de todo o conjunto. Para n, usou-se o seguinte conjunto $n = \{10, 20, 30, 40, 50, 100, 200, 300, 400, 500\}$ já para o tamanho percentual da mochila $PM = \{10\%, 20\%, 30\%, 40\%, 50\%\}$. Não bastando isso, gerou-se, para cada combinação de n e PM, dez instâncias. Além dessa combinação entre n e PM, tem-se também que são quatro os tipos de conjunto de instâncias propostos. Então, de fato, a combinação é entre n, PM e o tipo de conjunto de instâncias. Seja, então, o cálculo do número de instâncias geradas: 10 (n: número de objetos) $\times$ 5 (PM: tamanho percentual da mochila) $\times$ 4 (quantidade de tipos de conjunto de instâncias) $\times$ 10 (número de instâncias para cada combinação), perfazendo um total de 2000 instâncias. Dessa forma para cada combinação de n, PM e tipo de conjunto de instância, tem-se 10 instâncias para experimentos.

O processo de geração de uma instância é implementado pelo método *GenerateRandom*(MAX,*dt*), apresentado no Programa 4. Esse método da classe *CKnapSack* recebe como parâmetros dois valores, o primeiro, MAX, determina o valor máximo da faixa de valores, e o segundo, *dt*, determina o tipo de conjunto de instânicas a ser gerado. Nesse módulo/método, o membro *Key* da estrutura de dados *Object*, apresentado no Programa 1 é utilizado pelo método *GenerateKeyByRatio*(), o qual gera essa chave *Key* como sendo a razão entre a utilidade, *P*, o peso, *W* para ser ordenado pelo algoritmo de *Heapsort*. O último parâmetro da função *Heapsort* indica se os dados devem ser ordenados de forma ascendente (1) ou descendente (0). Os dados são gerados em ordem randômicas e depois ordenados pela razão, pelo bel-prazer do autor. O *Heapsort* foi escolhido, pois ele apresenta desempenho estável para qualquer caso, ou seja, ele é de fato, $O(n \log n)$. Essa estabilidade garante que o processamento dos algoritmos do problema não serão atrapalhados por eventuais diferenças de desempenho na ordenação de um conjunto de instânicas particular.

```
void CKnapSack::GenerateRandom(int MAX,int dt)
{
  int i;

  /* generating random numbers */
  /* initializing seed to randomness*/
  struct timeval seed;
  /* using time as seed for srand() function */
  gettimeofday(&seed,NULL);
  srand48((int)(seed.tv_sec + 1000000*seed.tv_usec));

  for( i = 1 ; i<=m_in ; i++)
    m_pData[i].W= (int)(uniformrand0to1()*(MAX-1) + 1 + 0.5);

  for( i = 1 ; i<=m_in ; i++)
```

```
  {
    switch(dt)
    {
    case 1:
    case 2:
      m_pData[i].P = (int)(uniformrand0to1()*(MAX-1) + 1 + 0.5);
      break;
    case 3:
      m_pData[i].P = m_pData[i].W + 10;
      break;
    case 4:
      m_pData[i].P = (int) (1.1 * m_pData[i].W + 0.5);
      /* +0.5 => to round */
      break;
    }
  }

  GenerateKeyByRatio();
  Heapsort(m_pData,&m_in,0);
}
```

Programa 4: Método *GenerateRandom*().

A função *gettimeofday*(), que retorna a hora do dia, é utilizada para inicializar o sistema de geração de números randômicos. A função que recebe a semente é *srand48*(). Esse procedimento de inicialização de semente é utilizado para evitar que os dados tenham algum padrão repetidamente aleatório, o que inviabilizaria a idéia de números aleatórios. Esse procedimento de inicialização é realizado a cada vez que uma nova instância é gerada.

```
int CSystem::ModuleGenerator()
{
  /* Generator */
  const int iCTE[] = {-1,1000,100,100,100};

  strcpy(sPathData,m_ppcargv[2]);
  parameter2(m_ppcargv[3],m_ids    ,m_idf    );
  parameter2(m_ppcargv[4],m_ins    ,m_inf    );
  parameter2(m_ppcargv[5],m_iMs    ,m_iMf    );
  parameter2(m_ppcargv[6],m_iInsts,m_iInstf);

  fprintf(stdout,"dataset - %d - %d\n"              ,m_ids    ,m_idf    );
  fprintf(stdout,"object's number - %d - %d\n"      ,m_ins    ,m_inf    );
  fprintf(stdout,"knapsack size - %d%% - %d%%\n"    ,m_iMs    ,m_iMf    );
  fprintf(stdout,"number of instances - %d - %d\n",m_iInsts,m_iInstf);

  for(dt=m_ids;dt<=m_idf;dt++)
  {
    b=(int)(log(m_ins)/log(10.));
    for(n=m_ins;n<=m_inf;n+=(int)pow(10.,b))
    {
      for(inst=m_iInsts;inst<=m_iInstf;inst++)
      {
        CKnapSack *m_poKnapSack = new CKnapSack();

        m_poKnapSack->Setn(n);
        m_poKnapSack->GenerateRandom(iCTE[dt],dt);
```

```
      for(dPM=m_iMs;dPM<=m_iMf;dPM+=10)
      {
        m_poKnapSack->SetPM(dPM);
        sprintf(sFileOut,"dset%1dn%03dM%03di%03d.dat"
           ,dt,n,(int)dPM,inst);
        sprintf(sTmp,"%s%s",sPathData,sFileOut);
        fprintf(stdout,"%s\n",sTmp);

        m_poKnapSack->FileSaveInstance(sTmp);
      }
      delete m_poKnapSack;
    }
    if ( (n / pow(10,b)) == 5)
    {
      n = 0;
      b++;
    }
  }
}

return 0; /* ok */
}
```

Programa 5: Gerador de Instânicas - Módulo *generator*.

Para gerar todas as instâncias foi desenvolvido um método, na verdade um módulo, de uma classe gerenciadora chamada *CSystem* que permite gerar todas essas 2000 instâncias de forma automática. Foi determinado um padrão para a composição do nome do arquivo das instâncias. No nome do arquivo são descritos todas as variáveis envolvidas no processo de geração, ou seja ele é definido da seguinte forma:

$$dset\%1dn\%03dM\%03di\%03d.dat$$

onde %1d, %03d, %03d e %03d serão sequencialmente substituídos pelo tipo de conjunto de instâncias, pelo número de objetos, n, pelo tamanho percentual da mochila PM, e, por ultimo, pelo número da instância.

Esse padrão gerado permitiu que essas instâncias fossem submetidas as várias abordagens e as soluções obtidas entre eles fossem comparadas.

Vale ressaltar que esse módulo está pronto a gerar muito mais instâncias por combinação com o objetivo de aumentar mais ainda o suporte estatístico dos experimentos. Nesse trabalho limitou-se somente as 2000 instâncias por questões de armazenamento restrito e tempo para análise dos resultados dos experimentos. Esse método é apresentado no Programa 5. Mais detalhes sobre o método *ModuleGenerator*() e sobre a classe *CSystem* serão apresentados na Seção 7.

5 Abordagem

Nesta seção são apresentadas algumas abordagens que resolvem o problema da mochila. A abordagem de *Backtracking* (Seção 5.1) e de Programação Dinâmica (Seção 5.2) obtém solução ótima, no entanto, têm execução em tempo exponencial. Já a abordagem Gulosa (Seção 5.2) têm tempo de execução linear, mas não garante a obtenção de solução ótima. Mais detalhes sobre os algoritmos, implementações, e análises de complexidades de tempo e de espaço são dados nas subseções seguintes.

5.1 Tentativa e Erro (*Backtracking*)

O espaço de solução para este problema consiste de 2^n maneiras distintas para atribuir valores zero ou um à x_i, com $1 \leq i \leq n$. Uma organização possível para o espaço de solução para este problema é uma organização por árvore. Um exemplo desse esquema é apresentado na Figura 2 como sendo uma árvore de número fixo de tuplas. Algoritmos de *Backtracking* podem usar esse espaço de estado para resolver o problema da mochila.

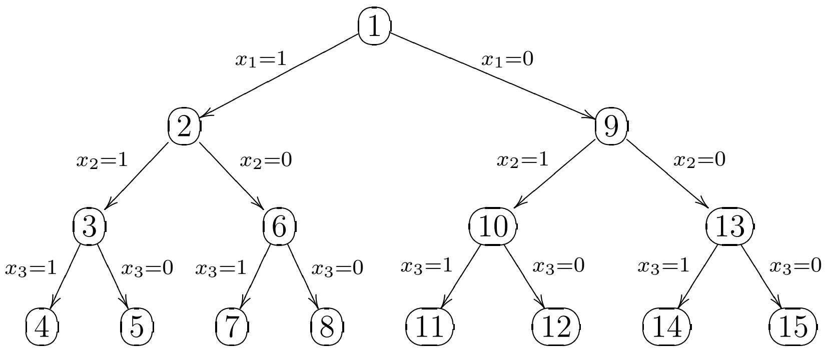


Figura 2: Uma árvore de espaço de estado com tuplas de tamanho fixo. Os nós são numerados de acordo com a busca em profundidade.

Uma função limitadora deve ser empregada para ajudar a eliminar alguns nós vivos sem a necessidade de realmente expandí-los. Uma boa função limite para este problema é obtida através do uso de um limite superior sobre o valor da melhor solução possível obtida pela expansão de um dado nó vivo e alguns de seus descendentes. Caso esse limite superior não é maior que o valor da melhor solução obtida até o momeneto, então o nó vivo pode ser eliminado.

Se em um nó Z os valores de x_i, $1 \leq i \leq k$ já foram determinados, então um limite superior para Z pode ser estimado através da relaxação da exigência de $x_i = 0$ ou $x_i = 1$ para $0 \leq x_i \leq 1$ e usando um algoritmo Guloso Ingênuo para resolver o problema relaxado, como o apresentado no Algoritmo 2.

O procedimento $Bound1(p, w, k, M)$, apresentado no Algoritmo 3 determina um limite superior sobre a melhor solução obtida através da expansão de qualquer nó Z no nível $k+1$ da árvore do espaço de estados. Os pesos e as utilidades do objeto i são w_i e p_i, respectivamente. $p = \sum_{i=1}^{k} p_i \times x_i$ e é assumido que $p_i/w_i \geq p_{i+1}/w_{i+1}$, $1 \leq i \leq n$, ou seja, os objetos estão ordenados pela razão entre a utilidade e o peso.

Do Algoritmo 3 segue que um limite para um possível filho esquerdo de um nó Z é o mesmo limite que para o próprio Z. Portanto, a função limitadora/limite não precisa ser computada sempre que o algoritmo *Backtracking* faz um movimento

Algorítmo 2 *GreedyKnapSack* - Uma solução Gulosa e Ingênua para resolver o problema da mochila relaxado, para valores reais de x_i, com $0 \leq i \leq 1$.

procedure $GreedyKnapSack(M, n, P, W, X)$
$\{M$, the knapsack size$\}$
$\{n$, the number of objects$\}$
$\{p_i \in P$ and $w_i \in W$ are the profit and weight of i object, respectively, $1 \leq i \leq n\}$
$\{$this objects must be sorted so that $p_i/w_i \geq p_{i+1}/w_{i+1}$ $\}$
$\{X(1..n)$, é o vetor solução$\}$
begin
$X \leftarrow 0$ $\{$initialize solution to zero$\}$
$cu \leftarrow M$ $\{cu =$ remaining knapsack capacity$\}$
$fp \leftarrow 0$ $\{fp =$ total profit$\}$
for $i \leftarrow 1$ to n **do**
  if $w_i > cu$ **then**
    break;
  end if
  $X(i) \leftarrow 1$
  $cu \leftarrow cu - w_i$
  $fp \leftarrow fp + p_i$
end for
if $i \leq n$ **then**
  $X(i) \leftarrow cu/w_i$
  $fp \leftarrow fp + p_i \times X(i)$
end if
return fp
end

Algorítmo 3 *Bound1* - Função limite para *BKnapSack1*.

procedure $Bound1(p, w, k, M)$
$\{p$, the current profit total$\}$
$\{w$, the current weight total$\}$
$\{k$, the index of the last removed item$\}$
$\{M$, the knapsack size$\}$
begin
$b \leftarrow p; c \leftarrow w$
for $i \leftarrow k + 1$ to n **do**
  $c \leftarrow c + w_i$
  if $c < M$ **then**
    $b \leftarrow b + p_i$
  else
    return $(b + (1 - (c - M)/w_i) \times p_i)$
  end if
end for
return b
end

para um filho esquerdo do nó. Visto que o algoritmo *Backtracking* vai tentar fazer um movimento para o nó filho esquerdo sempre que é dado uma escolha entre um nó filho esquerdo e direito, verifica-se, então, que a função limite precisa ser usada somente depois de uma série de movimentos para nós filhos esquerdo com sucesso (i.e. movimentos para filhos esquerdos possíveis). O resultado desse processo é apresentado no Algoritmo 4. Ele foi obtido através de um esquema iterativo de *Backtracking*, ou seja, Tentativa e Erro.

Uma versão mais enxuta e melhorada de $BKnapSack1$ (Algoritmo 4) pode ser obtida, a medida que o primeiro *loop* (*while*) de $BKnapSack1$ é essencialmente executado cada vez que uma chamada a $Bound1$ é feita. Portanto, não deveria haver a necesidade de refazer este trabalho. Para evitar que esse trabalho seja feito, precisa-se modificar a função limite $Bound1$ gerando uma função com efeito similar. Então, os novos algoritmos são apresentados nos Algoritmos 6 e 5. Todas as variáveis tem o mesmo significado que nos Algoritmos 4 e 3. Este dois algoritmos para a solução do problema da mochila e suas funções limites são propostos por Horowitz em [1]. Essa otimização de código apresentada pela segunda versão do algoritmo $BKnapSack2$ apresentou uma melhora no desempenho de cerca de 10% do tempo de execução, na Seção 6 isso será melhor discutido. Isso já era esperado, pois nenhuma redução no espaço de estados foi feita, simplesmente uma otimização de código.

As implementações para os algoritmos $BKnapSack1$ e $BKnapSack2$ estão nos Programas 7 e 9, respectivamente, e as implementações dos algoritmos $Bound1$ e $Bound2$ estão nos Programas 6 e 8, respectivamente.

```
double CKnapSack::Bound1(double p,double w,int k)
{
  /* p, the current profit total */
  /* w, the cuirrent weight total */
  /* k, the index of the last removed item */
  /* the result is a new profit */
  int i;
  double b,c;

  b = p;
  c = w;

  for(i = k+1 ; i <= m_in ; i++)
  {
    c += m_pData[i].W;
    if ( (c <= m_iM) )
      b += m_pData[i].P;
    else
      return (b + (1 - (c - m_iM)/m_pData[i].W)*m_pData[i].P);
  }
  return b;
}
```

Programa 6: Implementação da função de limite para $BKnapSack1$ para o problema da mochila.

```
void CKnapSack::BKnapSack1()
{
  int    k;
  double cw,cp;
```

Algorítmo 4 $BKnapSack1$ - Algoritmo de Backtracking para o problema da Mochila.

```
procedure BKnapSack1(M, n, fw, fp, X)
{M, the knapsack size}
{n, the number of weights and profit}
{fw, the final weight of knapsack}
{fp, the final maximum profit}
{X(1 : n), either 0 or 1, X(k) = 0 if w_k is not in knapsack else X(k) = 1}
{Y(1 : n) momentaneous/parcial solution}
begin
cw ← cp ← 0 {cw = current weight, cp = current profit}
k ← 1; fp ← −1
loop
  while k ≤ n and cw + w_k ≤ M do
    {place w_k into knapsack}
    cw ← cw + w_k
    cp ← cp + p_k
    Y(k) ← 1
    k ← k + 1
  end while
  if k > n then
    fp ← cp
    fw ← cw
    k ← n
    X ← Y {update solution}
  else
    Y(k) ← 0 {M is exceeded so object k does not fit}
  end if
  {after fp is set above, Bound1 = fp}
  while ⌊Bound1(cp, cw, k, M)⌋ ≤ fp do
    while k ≠ 0 and Y(k) ≠ 1 do
      k ← k − 1 {find the last weight included in the knapsack}
    end while
    if k = 0 then
      return {the algorithm ends here}
    end if
    {remove the kth item}
    Y(k) ← 0
    cw ← cw − w_k
    cp ← cp − p_k
  end while
  k ← k + 1
end loop
end
```

Algorítmo 5 *Bound2* - Função limite para *BKnapSack2* - Modificação.

procedure $Bound2(p, w, k, M, pp, ww, i)$
{pp and ww are the profit and weight corresponding to the last left child move.}
{i is the index of the first object that does not fit.}
{It if $n+1$ if no objects remain.}
begin
$pp \leftarrow p; ww \leftarrow w$
for $i \leftarrow k+1$ to n **do**
  if $ww + w_i \leq M$ **then**
    $ww \leftarrow ww + w_i$
    $pp \leftarrow pp + p_i$
    $Y(i) \leftarrow 1$
  else
    return $(pp + (M - ww) \times p_i/w_i)$
  end if
end for
return pp
end

```
StartProcess();
CreateDataSolution();
CreateDataBacktracking();

GenerateKeyByRatio();
Heapsort(m_pData,&m_in,0);

/* computing space to be used */
/* X(1..n), Y(1..n) */
m_iSpace = 2*m_in;
/* constante */

cw = 0.;
cp = 0.;
k  = 1;
m_ifp = -1;

while(!g_bOverflow)
{
  while( (k <= m_in) && ((cw + m_pData[k].W) <= m_iM) )
  {
    cw += m_pData[k].W;
    cp += m_pData[k].P;
    m_piY[k] = 1;
    k++;
  }

  if (k > m_in)
  {
    m_ifp = (int)cp;
    m_ifw = (int)cw;
    k = m_in;
    /* solucao encontrada */
```

Algorítmo 6 $BKnapSack2$ - Algoritmo de Backtracking para o problema da Mochila - Modificação.

```
procedure BKnapSack2(M, n, fw, fp, X)
{M, the knapsack size}
{n, the number of weights and profit}
{fw, the final weight of knapsack}
{fp, the final maximum profit}
{X(1 : n), either 0 or 1, X(k) = 0 if w_k is not in knapsack else X(k) = 1}
{Y(1 : n) momentaneous/parcial solution}
begin
cw ← cp ← 0 {cw = current weight, cp = current profit}
k ← 0; fp ← −1
loop
  while ⌊Bound2(cp, cw, k, M, pp, ww, j)⌋ ≤ fp do
    while k ≠ 0 and Y(k) ≠ 1 do
      k ← k − 1 {find the last weight included in the knapsack}
    end while
    if k = 0 then
      return {the algorithm ends here}
    end if
    {remove the kth item}
    Y(k) ← 0
    cw ← cw − w_k
    cp ← cp − p_k
  end while
  {equivalent to loop of first loop in BKnapSack1 (Algoritmo 4)}
  cw ← ww
  cp ← pp
  k ← j
  if k > n then
    fp ← cp
    fw ← cw
    k ← n
    X ← Y {update solution}
  else
    Y(k) ← 0 {M is exceeded so object k does not fit}
  end if
end loop
end
```

```
      m_bSolution = TRUE;
      /* X <- Y */
      for(int l=1;l<=m_in; l++)
        m_piX[l] = m_piY[l];
    }
    else
      m_piY[k] = 0;

    while( (int)Bound1(cp,cw,k) <= m_ifp)
    {
      while ( k != 0 && m_piY[k] != 1)
        k--;

      if (k == 0)
      {
        /* anula alarme de Overflow */
        alarm(0);

        m_bOptimal = TRUE;
        m_bOverflow = g_bOverflow;
        VerifyResult();
        FinishTime();
        return; /* final */
      }

      m_piY[k] = 0;
      cw -= m_pData[k].W;
      cp -= m_pData[k].P;
    }
    k++;
  }
  m_bOverflow = TRUE;
  FinishTime();
}
```

Programa 7: Implementação do primeiro algoritmo de *Backtracking* para o problema da mochila.

```
double CKnapSack::Bound2(double p,double w,int k,double& pp,double& ww,int& i)
{
  /* pp and ww are the profit and weight corresponding to the last left */
  /* child move, i is the index of the first object that doesn not fit. */
  /* It is n+1 if no objects remain. */
  pp = p;
  ww = w;
  for(i = k+1 ; i <= m_in ; i++)
  {
    if ( (ww + m_pData[i].W) <= m_iM )
    {
      ww += m_pData[i].W;
      pp += m_pData[i].P;
      m_piY[i] = 1;
    }
    else
    {
      return (pp + (m_iM - ww)*m_pData[i].P/m_pData[i].W);
    }
  }
```

```
  return pp;
}
```

Programa 8: Implementação da função de limite para $BKnapSack2$ para o problema da mochila.

```
void CKnapSack::BKnapSack2()
{
  int      k, j;
  double cw, cp, ww, pp;

  StartProcess();
  CreateDataSolution();
  CreateDataBacktracking();

  GenerateKeyByRatio();
  Heapsort(m_pData,&m_in,0);

  /* computing space to be used */
  /* X(1..n), Y(1..n) */
  m_iSpace = 2*m_in;
  /* constante */

  cw = 0.;
  cp = 0.;
  k  = 0;
  m_ifp = -1;

  while(!g_bOverflow)
  {
    while( (int)Bound2(cp,cw,k,pp,ww,j) <= m_ifp)
    {
      while ( k != 0 && m_piY[k] != 1)
        k--;

      if (k == 0)
      {
        /* anula alarme de Overflow */
        alarm(0);

        m_bOptimal = TRUE;
        m_bOverflow = g_bOverflow;
        VerifyResult();
        FinishTime();
        return; /* final */
      }

      m_piY[k] = 0;
      cw -= m_pData[k].W;
      cp -= m_pData[k].P;
    }

    cp = pp;
    cw = ww;
    k = j;

    if (k > m_in)
    {
```

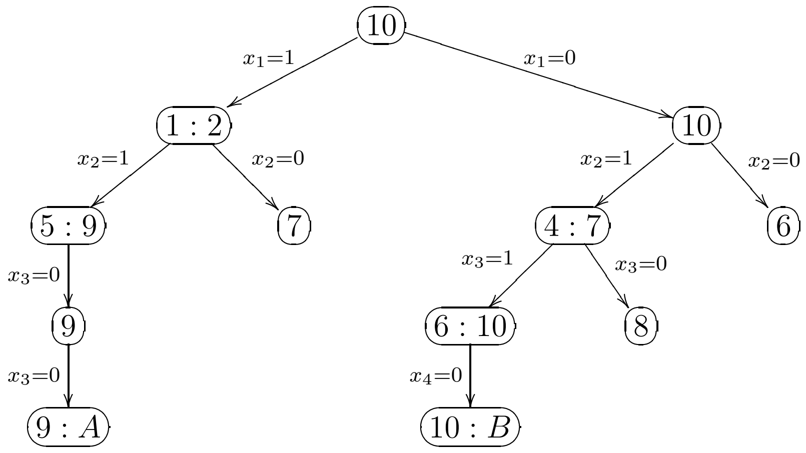
```
      m_ifp = (int)cp;
      m_ifw = (int)cw;
      k = m_in;
      /* found solution */
      m_bSolution = TRUE;
      /* X<- Y */
      for(int l=1;l<=m_in; l++)
        m_piX[l] = m_piY[l];
    }
    else
      m_piY[k] = 0;
  }
  m_bOverflow = TRUE;
  FinishTime();
}
```

Programa 9: Implementação do segundo algoritmo de *Backtracking* para o problema da mochila.

Seja o seguinte exemplo, para ilustrar o funcionamento do algoritmo de *Backtracking*: o número de objetos $n = 4$, o tamanho da mochila $M = 6$, e $(W, P) = \{(1, 2), (2, 3), (4, 7), (5, 4)\}$.

A abordagem *Backtracking* exige que os pares (W, P) estejam ordenados pela razão entre P e W, então seja o conjunto de pares $(W, P) = \{(1, 2), (4, 7), (2, 3), (5, 4)\}$, ordenado pela razão.

Figura 3: Exemplo do algoritmo *Backtracking*. $n = 4$, $M = 6$, e $(W, P) = \{(1, 2), (4, 7), (2, 3), (5, 4)\}$. A solução é apontado por $10 : B$, com $fp = 10$ e $fw = 6$, e o conjunto solução $X = \{0, 1, 1, 0\}$ ou $Sol = \{(4, 7), (2, 3)\}$.



A Figura 3 ilustra a solução do exemplo citado acima. A figura pode ser melhor explicada: quando um número aparece isolado, significa o valor retornado pela função limite (*Bound1*(), ou *Bound2*()); quando dois números aparecem em um nó separados pelo símbolo : $(cw : cp)$, os números indicam o peso e a utilidade acumulados naquele nó; quando um número e uma letra aparecem separados pelo símbolo : $(fp : Z)$, indica que uma solução foi encontrada. A solução que tiver o maior fp, ou seja, a última solução encontrada pelo algoritmo corresponde a solução ótima.

Dos $2^{n+1} - 1 = 2^5 - 1 = 32$ nós no espaço de estados da árvore, somente 12 foram gerados. Neste exemplo não foi apresentada redução devido à função chão

- *floor* ($\lfloor \bullet \rfloor$) aplicada a função limite (*Bound1*() ou *Bound2*()). Mas, no entanto, essa função não permite que nós que apresentem utilidade superiores em menos de uma unidade à utilidade máxima encontrada até o instante, fp, sejam estendidos. Esses, não retornarão utilidade superior ao máximo até o momento. Isto porque as utilidades consideradas neste trabalho são inteiras. Um breve comentário sobre isso é cabível aqui. Com essa simples imposição da função chão o tempo total de computação, das 2000 instâncias que são consideradas para experimentos, instâncias caiu em certa de 50%.

Nesta seção apresentou-se a abordagem *Backtracking* ou Tentativa e Erro para solução do problema da mochila. Na próxima subseção será abordada a abordagem de Programação Dinâmica.

5.2 Programação Dinâmica (*Dynamic Programming*)

A Programação Dinâmica é uma metodologia de desenvolvimento de algoritmos que pode ser usada quando a solução de um problema pode ser vista como um resultado de uma seqüência de decisões [1]. Uma maneira eficiente de se resolver o problema da mochila é por meio de uma técnica conhecida por Programação dinâmica [5]. Esta técnica consiste em resolver o problema via decomposição em subproblemas, solução desses subproblemas e a posterior combinação das respostas obtidas. Um exemplo conhecido desta técnica é o algoritmo *Quicksort*, que ordena um conjunto fazendo sua decomposição em subconjuntos, que serão posteriormente ordenados pelo mesmo algoritmo aplicado recursivamente.

$$\begin{aligned} max \sum_{i=l}^{j} x_i \times p_i \\ \text{sujeito a } \sum_{i=l}^{j} x_i \times w_i \leq M \\ x_i = 0 \text{ ou } 1, l \leq i \leq n \end{aligned} \tag{2}$$

Seja o problema da mochila 0/1 representado por $Knap(1, n, Y)$. Seja, também, $y_1, y_2, \ldots, y_n$ uma seqüência ótima de valores 0/1 para $x_1, x_2, \ldots, x_n$, respectivamente. Se $y_1 = 0$ então $y_2, y_3, \ldots, y_n$ deve constituir uma seqüência ótima para o problema $Knap(2, n, Y)$, Se ele não for, então $y_1, y_2, \ldots, y_n$ não é uma seqüência ótima para $Knap(1, n, Y)$. Se $y_1 = 1$ então $y_2, y_3, \ldots, y_n$ deve ser uma seqüência ótima para o problema $Knap(2, n, Y)$. Se ela não for, então existe uma outra seqüência de 0/1 de $z_2, z_3, \ldots, z_n$ tal que $\sum_{i=2}^{n} w_i \times z_i \leq M - w_1$ e $\sum_{i=2}^{n} p_i \times z_i > \sum_{i=2}^{n} p_i \times y_i$. Portanto, a seqüência $y_1, z_2, z_3, \ldots, z_n$ é uma seqüência para a equação 2 com o maior valor. Assim, aplica-se o princípio da otimalidade.

Seja $g_j(y)$ o valor de uma solução ótima $Knap(j + 1, n, y)$. Claramente, $g_0(M)$ é o valor de uma solução ótima $KNAP(1, n, M)$. As decisões possíveis para x_1 são 0 e 1 ($D_1 = \{0, 1\}$). Do princípio da otimalidade, segue que:

$$g_0(M) = max\{g_1(M), g_1(M - w_1) + p_1\} \tag{3}$$

Enquanto que o princípio da otimalidade foi definido somente para os estados e decisões iniciais, ele deve ser aplicado igualmente para os estados e decisões intermediárias.

Então, seja $y_1, y_2, \ldots, y_n$ uma seqüência ótima de valores 0/1 para $Knap(1, n, M)$. Para cada j, $1 \leq j \leq n$, $y_1, y_2, \ldots, y_j$ e $y_{j+1}, \ldots, y_n$ devem ser soluções ótimas para $Knap(1, j, \sum_{i=1}^{j} w_i \times y_i)$ e $Knap(j+1, n, M - \sum_{i=1}^{j} w_i \times y_i)$, respectivamente. Essa observação permite generalizar a Equação 3 para:

$$g_i(y) = max\{g_{i+1}(y), g_{i+1}(y - w_{i+1}) + p_{i+1}\} \tag{4}$$

A aplicação recursiva do princípio da otimalidade resulta em uma relação de recorrência do tipo 4. O Algoritmo de Programação Dinâmica resolve esta recorrência para obter uma solução para uma dada instância do problema. A recorrência da equação 4 pode ser resolvida usando $g_n(y) = 0$ para todo y. De $g_n(y)$ pode-se obter $g_{n-1}(y)$ usando a Equação 4 com $i = n - 1$. Então, usando $g_{n-1}(y)$ pode-se obter $g_{n-2}(y)$. Aplicando-se essa recursividade, pode-se determinar $g_1(y)$ e finalmente $g_0(M)$ usando a Equação 4, com $i = 0$.

O problema da mochila, definido pela Equação 1 pode ser representado pelo Algoritmo 7. Note que esse algoritmo mostra o inverso do apresentado pela Equação 4, de fato o inverso e correto é o apresentado pela Equação 5. Ele parte de $g_n(M)$. Ou seja, os problemas são subdividos de g_{n-1}, g_{n-2} até $g_0(y)$. Dessa forma ele parte de $g_0(X) = 0$ para todos X e $g_i(x) = -\infty$, com $x < 0$. Os valores de $g_1, g_2, \ldots, g_n$ são sucessivamente calculados usando a Equação 5.

$$g_i(X) = max\{g_{i-1}(X), g_{i-1}(X - w_i) + p_i\} \tag{5}$$

Algorítmo 7 *DKP* - Algoritmo informal de Programação Dinâmica para o problema da mochila.

procedure $DKP(p, w, n, M)$
{p is the set of profits}
{w is the set of weights}
{n is the number of objects}
{M is knapsack size}
begin
$S^0 \leftarrow \{(0, 0)\}$
for $i \leftarrow 1$ to $n - 1$ **do**
  $S_1^i \leftarrow \{(W1, P1)|(W1 - w_i, P1 - p_i) \in S^{i-1}$ and $W1 \leq M\}$
  $S^i \leftarrow MergePurge(S^{i-1}, S_1^i)$
end for
$TraceBack()$
end

O Algoritmo 7 pode ser descrito, superficialmente da seguinte forma. Um estado inicial, S^0 definido com o par $(W, P) \leftarrow (0, 0)$. As tuplas ou pares do estado, S^i, determinam as possíveis situações do problema quando os i primeiros objetos são analisados. Para determinar o estado, S^i a partir de um estado S^{i-1} deve-se a princípio computar $g_{i-1}(X - w_i) + p_i$. Esse computo é realizado adicionando-se a cada par de S^{i-1} o par (w_i, p_i), ou seja:

$$S_1^i = \{(W, P)|(W - w_i, P - p_i) \in S^{i-1}\}$$

S^i pode agora ser obtido agrupando S^{i-1} e S_1^i. Esse agrupamento realiza o processo de pegar o máximo entre $g_{i-1}(X)$ e $g_{i-1}(X - w_i) + p_i$ na Equação 5. Dessa forma, se um par de S^{i-1} ou S_1^i tem um par (W_j, P_j) e o outro tem um par (W_k, P_k) com $P_j \leq P_k$ e $W_j \geq W_k$, então o par (P_j, W_j) deve ser descartado.

Um algoritmo contendo estes passos com mais detalhes pode ser visto no Algoritmo 8. Já a implementação desse Algoritmo 8 pode ser vista no Programa 10.

O algoritmo $DKnapSack1$, apresentado em 8, pode ser otimizado com o uso de algumas heurísticas sugeridas por Horowitz [1]. Seja L uma estimativa do valor de uma solução ótima tal que $g_n(M) \geq L$. Seja $PLeft(i) = \sum_{j=i+1}^{n} p_j$. Se S^i contém uma tupla (W, P) tal que $P + PLeft(i) < L$ então (W, P) deve ser removido (*purged*) de S_i. Para ver isso, observe que (W, P) pode contribuir no máximo com o par $(W + \sum_{j=i+1}^{n} w_j, P + \sum_{j=i+1}^{n} p_j)$ para S_1^n. Como $P + \sum_{j=i+1}^{n} p_j = P + PLeft(i) < L$, tem-se que esse par não pode levar a no mínimo L e então ele não pode determinar um solução ótima.

Uma forma simples de estimar L de forma que $L \leq g_n(M)$ é considerar o último par (W, P) em S^i. Então, $P \leq g_n(M)$. Uma melhor estimativa é obtida adicionando-se a (W, P) alguns dos objetos restantes, respeitando o tamanho da mochila M.

Um algoritmo que implementa essas heurísticas visando otimização para a abordagem de Programação Dinâmica pode ser visto no Algoritmo 9. A implementação desse Algoritmo 9 com as otimizações (usando heurísticas citadas) pode ser vista no Programa 11. Uma prévia sobre os experimentos: essa otimização fez com que o tempo de processamento (e o espaço requerido também) da grande maioria das instâncias, conjuntos de instâncias 1, 2 e 4, caísse pela metade, na média, para o Algoritmo 9 (implementado no Programa 11) em relação ao Algoritmo 8 (implementado no Programa 10).

Ambas implementações, alocam memória dinâmicamente, a medida da demanda, e também realocam o espaço não utilizado, evitando, dessa forma, a fragmentação / segmentação da memória. Em Horowitz [1] é proposto a utilização de dois vetores para o armazenamento das produções S^i. Isso requer que este vetor seja alocado no início da execução do programa.

Aqui, implementa-se um esquema de alocação de memória para a produção S^i, com base na cardinalidade da produção S^{i-1}. Sabe que a cardinalidade de S^i, no pior caso, é 2 vezes a cardinalidade de S^{i-1}, ou seja, $|S^i| = 2|S^{i-1}|$. Ao final da iteração i, sabe-se, de fato, quantos estados ou tuplas existem na produção $|S^i|$. Com essa informação, a memória que foi alocada e não foi utilizada é liberada. Esse processo é feito com o comando *realloc*, que permite que um bloco de memória seja realocado, mantendo, intocável, uma parte relevante. Essa produção S^i é implementada pela primitiva *m_ppData* da classe *CKnapSack*. A qual, já citada anteriormente, é um vetor de ponteiros, ou ponteiros de *Vector*'s para tipos de dados *Objects*.

Sabe-se que a exigência de espaço para o problema da mochila, utilizando-se essa implementação do Algoritmo 7 de Programação Dinâmica, é de $O(min\{2^n, n\sum_{i=1}^{n} p_i, nM\})$, ou seja, depende predominantemente do mínimo entre 2^n (o exponencial do número de objetos), $n\sum_{i=1}^{n} p_i$ (produto entre o número de objetos e o somatório das utilidades dos objetos) e nM (produto entre o número de objetos e a capacidade da mochila). Maiores detalhes sobre essa análise podem ser encontrados em [1], página 226. A mesma análise pode ser estendida para a complexidade de tempo. No entanto, pode-se dizer que para as instâncias propostas neste trabalho, o fator nM é dominante entre os outros, ou seja, ele é mínimo entre os outros dois valores, haja visto que os valores de peso e utilidade são inteiros.

```
void CKnapSack::DKnapSack1()
```

Algorítmo 8 $DKnapSack1$ - Algoritmo Detalhado de Programação Dinâmica para o problema da mochila.

procedure $DKnapSack1(p, w, n, M)$
{p is the set of profits}
{w is the set of weights}
{n is the number of objects}
{M is knapsack size}
{S^i is the set of all pair (W, P) representing i state}
begin
$S^0(1) \leftarrow (0, 0)$
for $i \leftarrow 1$ to $n - 1$ **do**
  $k \leftarrow 1$ {related with S^{i-1} }
  $m \leftarrow 1$ {related with S^i }
  $u \leftarrow$ largest $k, 1 \leq k \leq |S^{i-1}|$, such that $S^{i-1}(k).W + w_i \leq M$
  for $j = 1$ to u **do**
    $(ww, pp) \leftarrow S^{i-1}(j) + (w_i, p_i)$ {next element in S^i_1}
    while $k \leq |S^{i-1}|$ and $S^{i-1}(k).W \leq ww$ **do**
      $S^i(m) \leftarrow S^{i-1}(k)$ {merge in from S^{i-1}}
      $m \leftarrow m + 1$; $k \leftarrow k + 1$
    end while
    if $k \leq |S^{i-1}|$ and $S^{i-1}(k).W = ww$ **then**
      $pp \leftarrow max(pp, S^{i-1}(k).P)$
      $k \leftarrow k + 1$
    end if
    if $pp > S^i(m - 1).P$ **then**
      $S^i(m) \leftarrow (ww, pp)$
      $m \leftarrow m + 1$
    end if
    while $k \leq |S^{i-1}|$ and $S^{i-1}(k).P \leq S^i(m - 1)$ **do**
      $k \leftarrow k + 1$ {purge}
    end while
  end for
  {merge in remaining terms from S^{i-1}}
  while $k \leq |S^{i-1}|$ **do**
    $S^i(m) \leftarrow S^{i-1}(k)$
    $m \leftarrow m + 1$;$k \leftarrow k + 1$
  end while
end for
$TraceBack()$
end

Algorítmo 9 *$DKnapSack2$* - Algoritmo Detalhado de Programação Dinâmica para o problema da mochila com otimização/heurísticas.

procedure $DKnapSackK2(p, w, n, M)$
{p is the set of profits}
{w is the set of weights}
{n is the number of objects}
{M is knapsack size}
{S^i is the set of all pair (W, P) representing i state}
begin
$L(1) \leftarrow GreedyKnapSack(1, n)$ {an initial estimative from any Greedy approach}
{Estimating PLeft(i), PLeft(n) = 0}
for $i = n-1$ to 1 step -1 **do**
  $PLeft(i) \leftarrow PLeft(i+1) + p_{i+1}$
end for
$S^0(1) \leftarrow (0, 0)$
for $i \leftarrow 1$ to $n-1$ **do**
  $k \leftarrow 1$ {related with S^{i-1} }
  $m \leftarrow 1$ {related with S^i }
  $u \leftarrow$ largest $k, 1 \leq k \leq |S^{i-1}|$, such that $S^{i-1}(k).W + w_i \leq M$
  for $j = 1$ to u **do**
    $(ww, pp) \leftarrow S^{i-1}(j) + (w_i, p_i)$ {next element in S_1^i}
    while $k \leq |S^{i-1}|$ and $S^{i-1}(k).W \leq ww$ **do**
      $S^i(m) \leftarrow S^{i-1}(k)$ {merge in from S^{i-1}}
      $m \leftarrow m+1$; $k \leftarrow k+1$
    end while
    if $k \leq |S^{i-1}|$ and $S^{i-1}(k).W = ww$ **then**
      $pp \leftarrow max(pp, S^{i-1}(k).P)$
      $k \leftarrow k+1$
    end if
    if $pp > S^i(m-1).P$ **then**
      $S^i(m) \leftarrow (ww, pp)$
      $m \leftarrow m+1$
    end if
    while $k \leq |S^{i-1}|$ and $S^{i-1}(k).P \leq S^i(m-1)$ and $S^{i-1}(k).P + PLeft(i) \leq L(i)$ **do**
      $k \leftarrow k+1$ {purge}
    end while
  end for
  {merge in remaining terms from S^{i-1}}
  while $k \leq |S^{i-1}|$ **do**
    $S^i(m) \leftarrow S^{i-1}(k)$
    $m \leftarrow m+1$;$k \leftarrow k+1$
  end while
  {Updating L - adding remains objects - pairs (W, P) }
  $L(i+1) \leftarrow S^i(|S^i|).P + GreedyKnapSack(i+1, n)$
end for
$TraceBack()$
end

```
{
  int k,i,j;
  double pp,ww;

  StartProcess();
  CreateDataSolution();
  CreateDataDynamicProgramming();

  GenerateKeyByRatio();

  Heapsort(m_pData,&m_in,0);

  AddPW(0,0,0); /* S^0 */
  for(i = 1 ; i<=m_in - 1 ; i++) /* generate S^i */
  {
    if (g_bOverflow) break;

    /* memory reallocation */
    m_piSizeM[i-1] = m_piSize[i-1]+1;
    m_ppData[i-1] =
      (Object*)realloc((void*)m_ppData[i-1],sizeof(Object)*m_piSizeM[i-1]);

    k = 1;
    /* u <- largest k, l <= k<=h, such that W(k) + wi <=M */
    for(j = 1 ;
      ( (j <= m_piSize[i-1]) && (m_ppData[i-1][j].W + m_pData[i].W<=m_iM) ) ; j++)
    {
      if (m_ppData[i-1][j].W + m_pData[i].W > m_iM)
        continue;

      /* next elemenst in S^i_1 */
      pp = m_ppData[i-1][j].P + m_pData[i].P;
      ww = m_ppData[i-1][j].W + m_pData[i].W;

      while( (k <= m_piSize[i-1]) &&
             (m_ppData[i-1][k].W<=ww ) ) /* merge in from S^{i-1} */
      {
        AddPW(i,m_ppData[i-1][k].P,m_ppData[i-1][k].W);
        k++;
      }

      if ( (k <= m_piSize[i-1]) && (m_ppData[i-1][k].W==ww) )
      {
        pp = max(pp,m_ppData[i-1][k].P);
        k++;
      }

      if ( (pp > m_ppData[i][m_piSize[i]].P) )
        AddPW(i,(int)pp,(int)ww);

      /* purge (skip) */
      while( (k <= m_piSize[i-1]) &&
             (m_ppData[i-1][k].P <= m_ppData[i][m_piSize[i]].P ) )
        k++;
    }

    /* merge in remaining terms from S^{i-1} */
    while( (k <= m_piSize[i-1]) )
```

```
    {
      AddPW(i ,m_ppData[i −1][k].P,m_ppData[i −1][k].W);
      k++;
    }
  }

  /* off overflow alarm */
  alarm(0);

  if (!g_bOverflow)
  {
    /* CallParts */
    TraceBack();
    VerifyResult();

    m_bOptimal    =TRUE;
    m_bSolution = TRUE;
    m_bOverflow = g_bOverflow;

    /* computing used Space */
    /* P(1..n), W(1..n), X(1..n), k */
    m_iSpace = 3*m_in;
    /* (P,W) 2*sum_{i=1}^{n−1}{|S^i|} */
    for(i=0;i<=m_in−1;i++)
      m_iSpace += 2*(m_piSize[i] + 1);
  }
  FinishTime();
}
```

Programa 10: Implementação do algoritmo de Programação Dinâmica para o problema da mochila.

```
void CKnapSack::DKnapSack2()
{
  int k,i,j;
  double pp,ww;

  StartProcess();
  CreateDataSolution();
  CreateDataDynamicProgramming();

  GenerateKeyByProfit();
  Heapsort(m_pData,&m_in,0);

  /* estimating a inicial L */
  ww = 0;
  m_piPL[1] = 0;
  for(i = 1 ; i <= m_in ; i++)
    if ( (ww + m_pData[i].W < m_iM) )
    {
      ww += m_pData[i].W;
      m_piPL[1] += m_pData[i].P;
    }

  /* estimating Pleft(i) */
  for(i=m_in−1;i>=0;i−−)
  {
    m_piPLeft[i] = m_piPLeft[i+1]+m_pData[i+1].P;
```

```
  }

  AddPW(0,0,0); /* S^0 */
  for(i = 1 ; i <=m_in - 1; i++) /* generate S^i */
  {
    if (g_bOverflow) break;

    /* memory reallocation */
    m_piSizeM[i-1] = m_piSize[i-1]+1;
    m_ppData[i-1] =
      (Object*)realloc((void*)m_ppData[i-1],sizeof(Object)*m_piSizeM[i-1]);

    /* create memory space for S^i and S^1_i */
    CreateVectorPW(i);

    k = 1;
    /* u <- largest k, l <= k<=h, such that W(k) + wi <=M */
    for(j = 1 ;
      ( (j <= m_piSize[i-1]) && (m_ppData[i-1][j].W + m_pData[i].W<=m_iM) ) ; j++)
    {
      if (m_ppData[i-1][j].W + m_pData[i].W > m_iM)
        continue;

      /* next elemenst in S^i_1 */
      pp = m_ppData[i-1][j].P + m_pData[i].P;
      ww = m_ppData[i-1][j].W + m_pData[i].W;

      while( (k <= m_piSize[i-1]) /* h = m_piSize[i-1] */&&
             (m_ppData[i-1][k].W<=ww) ) /* merge in from S^{i-1} */
      {
        AddPW(i,m_ppData[i-1][k].P,m_ppData[i-1][k].W);
        k++;
      }

      if ( (k <= m_piSize[i-1]) && (m_ppData[i-1][k].W==ww) )
      {
        pp = max(pp,m_ppData[i-1][k].P);
        k++;
      }
      /* m_ppiP[i][0] deve ser igual a zero */
      if ( (pp > m_ppData[i][m_piSize[i]].P) )
        AddPW(i,(int)pp,(int)ww);

      /* purge (skip) */
      while( (k <= m_piSize[i-1]) &&
             ((m_ppData[i-1][k].P <= m_ppData[i][m_piSize[i]].P) ||
              (m_ppData[i-1][k].P + m_piPLeft[i] < m_piPL[i])) )
                /* L - otimizacao, heuristica */
        k++;
    }

    /* merge in remaining terms from S^{i-1} */
    while( (k <= m_piSize[i-1]) )
    {
      AddPW(i,m_ppData[i-1][k].P,m_ppData[i-1][k].W);
      k++;
    }
    /* uptading L */
```

```
    ww = m_ppData[i][m_piSize[i]].W;
    m_piPL[i+1] = m_ppData[i][m_piSize[i]].P;
    for(j=i+1;j<=m_in;j++)
      if ( (ww + m_pData[j].W<=m_iM) )
      {
        ww += m_pData[j].W;
        m_piPL[i+1] += m_pData[j].P;
      }
  }

  /* off overflow alarm */
  alarm(0);

  if (!g_bOverflow)
  {
    /* CallParts */
    TraceBack();

    VerifyResult();

    m_bOptimal    = TRUE;
    m_bSolution = TRUE;
    m_bOverflow = g_bOverflow;

    /* computing used space */
    /* P(1..n), W(1..n), X(1..n), L(1..n), PLeft(1..n), k */
    m_iSpace = 5*m_in;
    /* (P,W) 2*sum_{i=1}^{n-1}{|S^i|} */
    for(i=0;i<=m_in-1;i++)
      m_iSpace += 2*(m_piSize[i] + 1);
  }
  FinishTime();
}
```

Programa 11: Implementação do algoritmo de Programação Dinâmica para o problema da mochila com o uso de heurísticas.

Seja o seguinte exemplo, para ilustrar o funcionamento do algoritmo de Programação Dinâmica: o número de objetos $n = 4$, o tamanho da mochila $M = 6$, e $(W, P) = \{(1,2), (2,3), (4,7), (5,4)\}$.

Caso esse exemplo for resolvido pela primeira abordagem, ordenado pela razão entre P e W, $(W, P) = \{(1,2), (4,7), (2,3), (5,4)\}$, tem-se as seguintes produções:

$$
\begin{array}{ll}
S^0 = \{(0,0)\} & S_1^1 = \{(1,2)\} \\
S^1 = \{(0,0), (1,2)\} & S_1^2 = \{(4,7), (5,9)\} \\
S^2 = \{(0,0), (1,2), (4,7), (5,9)\} & S_1^3 = \{(2,3), (3,5), (6,10)\} \\
S^3 = \{(0,0), (1,2), (2,3), (3,5), (4,7), (5,9), (6,10)\} &
\end{array}
$$

Caso o exemplo seja resolvido pela segunda abordagem, ordenado pela utilidade P, $(W, P) = \{(4,7), (5,4), (2,3), (1,2)\}$, tem-se as seguintes produções:

$$
\begin{array}{ll}
S^0 = \{(0,0)\} & S_1^1 = \{(4,7)\} \\
S^1 = \{(4,7)\} & S_1^2 = \emptyset \\
S^2 = \{(4,7)\} & S_1^3 = \{(6,10)\} \\
S^3 = \{(4,7), (6,10)\} &
\end{array}
$$

Uma comparação entre a quantidade de estados gerados pela primeira abordagem (14) contra os da segunda abordagem (5), permite mostrar o quão eficiente é o uso de heurísticas para otimizar a abordagem de Programação Dinâmica. Essas ordenações impostas para as abordagens foram as que resultaram em melhor desempenho de tempo de execução para todo o conjunto de 2000 instâncias.

Algorítmo 10 $TraceBack$ - Obtém a solução da abordagem de Programação Dinâmica, através das produções de estado S^i.

procedure $TraceBack()$
{M is knapsack size}
{S^i takes all tuples of state i}
{w_i and p_i are the weight and profit of i object, respectively}
{X is a 0/1 vetor contain problem solution}
begin
$(WX, WY) \leftarrow$ last tuple in S^{n-1}
$MX \leftarrow M$
for $k = n$ to 1 step -1 **do**
  $(WY, PY) \leftarrow (W1 + w_k, P1 + p_k)$ {where $W1$ is the largest W in any tuple in S^{k-1} such that $W + w_k \leq MX$}
  if $(PX > PY)$ or $((PX = PY)$ and $(WX \neq WY))$ **then**
    $X[k] \leftarrow 0$
  else
    $X[k] \leftarrow 1$
    $(WX, PX) \leftarrow (WY, PY) - (w_k, p_k)$
    $MX \leftarrow WY - w_k$
    if (MX = 0) **then**
      {the end}
      for $i = k - 1$ to 1 step -1 **do**
        $X[i] \leftarrow 0$
      end for
      return X;
    end if
  end if
end for
return X;
end

O algoritmo para realizar o processo de $TraceBack$ ou $CallParts$, conforme chamado pelo algoritmo $DKnapSack$ de Horowitz em [1], página 225, é apresentado no Algoritmo 10. A implementação desse algoritmo, apresentada no Programa 12 não possui ordem de complexidade de tempo $O(nmax_i\{\log|S^i|\}) = O(n^2)$ conforme sugerido na referência. A implementação possui ordem de complexidade de tempo de $O(n\sum_i^n |S^i|) = O(n2^n)$. Isso porque ao invés de utilizar uma busca binária $(O(\log|S^i|))$, utilizou-se uma busca simples $(O(|S^i|))$. No entanto, para as instâncias utilizadas, a implementação encontrou soluções em tempo razoável (menos de 1 segundos).

Para ilustrar o funcionamento do algoritmo $TraceBack()$, resolve-se o exemplo citado acima, dada as suas produções geradas pelo algoritmo $DKnapSack1$:

$S^0 = \{(0,0)\}$
$S^1 = \{(0,0),(1,2)\}$
$S^2 = \{(0,0),(1,2),(4,7),(5,9)\}$
$S^3 = \{(0,0),(1,2),(2,3),(3,5),(4,7),(5,9),(6,10)\}$

$S_1^1 = \{(1,2)\}$
$S_1^2 = \{(4,7),(5,9)\}$
$S_1^3 = \{(2,3),(3,5),(6,10)\}$

A princípio, tem-se $(WX, PX) \leftarrow (6,10)$ e $MX \leftarrow 6$. Iniciando a execução do *loop*:

- para $k = 4$, $(WX, PX) = (6,10)$ e $(WY, PY) = (6,6)$. Como a condição $(PX > PY)$ é verdadeira, então $X[4] \leftarrow 0$.
- para $k = 3$, $(WX, PX) = (6,10)$ e $(WY, PY) = (6,10)$. Como a condição $(PX > PY)$ é falsa, então $X[3] \leftarrow 1$, $(WX, PX) \leftarrow (6,10) - (2,3) = (4,7)$ e $MX \leftarrow 6 - 2 = 4$.
- para $k = 2$, $(WX, PX) = (4,7)$ e $(WY, PY) = (0,0)$. Como a condição $(PX > PY)$ é falsa, então $X[2] \leftarrow 1$. $(WX, PX) \leftarrow (4,7) - (4,7) = (0,0)$ e $MX \leftarrow 4 - 4 = 0$, como $MX = 0$, então, $X[1] \leftarrow 0$ e término.

Portanto, a solução para o problema definido como: $n = 4$, $M = 6$, $(W,P) = \{(1,2),(4,7),(2,3),(5,4)\}$ é $X = \{0,1,1,0\}$ ou $Sol = \{(4,7),(2,3)\}$.

```
void CKnapSack::TraceBack()
{
  int i,k;
  int PX,WX,PY,WY,MX;

  /* last tuple in S^{n-1} */
  PX = m_ppData[m_in-1][m_piSize[m_in-1]].P;
  WX = m_ppData[m_in-1][m_piSize[m_in-1]].W;
  MX = m_iM;

  /* loop */
  for(k=m_in;k>=1;k--)
  {
    PY = 0;
    WY = 0;
    for(i = 1 ; i <= m_piSize[k-1] ; i++)
    {
      if ( (m_ppData[k-1][i].W + m_pData[k].W<=MX) &&
           (m_ppData[k-1][i].W + m_pData[k].W>=WY) )
      {
        if (PY > m_ppData[k-1][i].P + m_pData[k].P)
          continue;
        PY = m_ppData[k-1][i].P + m_pData[k].P;
        WY = m_ppData[k-1][i].W + m_pData[k].W;
      }
    }
    if ( (PX > PY) || ( (PX==PY) && (WX != WY) ) )
    {
      /* updating fp and fw */
      if (k == m_in)
      {
        m_ifp = PX;
```

```
          m_ifw = WX;
        }
        m_piX[k] = 0;
      }
      else
      {
        /* updating fp and fw */
        if (k==m_in)
        {
          m_ifp = PY;
          m_ifw = WY;
        }
        m_piX[k] = 1;
        /* updatding PX e WX e MX */
        PX = PY - m_pData[k].P;
        WX = WY - m_pData[k].W;
        MX = WY - m_pData[k].W;
        /* in case WX== 0,it's ready */
        if (WX== 0)
        {
          for(i=k-1;i>=1;i--)
            m_piX[i] = 0;
          break;
        }
      }
    }
}
```

Programa 12: Implementação do algoritmo $TraceBack()$.

Nesta seção apresentou-se a abordagem de Programação Dinâmica para solução do problema da mochila. Na próxima subseção será abordada a abordagem Gulosa, ou abordagem *Greedy*.

5.3 Gulosa (*Greedy*) ou Heurística

Algoritmos Gulosos são tipicamente usados para resolver problemas de otimização. Considere o problema da mochila que tem como objetivo maximizar a utilidade dos objetos escolhidos respeitando uma restrição de peso M. Um algoritmo Guloso escolhe o objeto que parece mais promissor em qualquer instante, e nunca reconsidera essa decisão, independente do que possa acontecer mais tarde. Não existe necessidade de avaliar alternativas, nem de empregar procedimentos sofisticados que permitam desfazer decisões tomadas previamente. A razão do algoritmo ser chamado Guloso é que o algoritmo escolhe, a cada passo, o candidato mais evidente que possa ser adicionado à solução.

Considere o seguinte problema geral: a partir de um conjunto C deseja determinar um subconjunto $S \subseteq C$ tal que (i) S satisfaça uma dada propriedade P, e (ii) S é mínimo (ou máximo) em relação a algum critério α. Ou seja, S é o menor (ou maior) subconjunto de C, segundo α, que satisfaz P. O algoritmo Guloso para resolver o problema geral consiste em um processo iterativo em que S é construído adicionando-se ao mesmo elementos de C um a um. Algoritmos Gulosos e os problemas que eles conseguem resolver são caracterizados pelas seguintes considerações:

- Considere um problema em que a solução ótima deve ser obtida. Para construir a solução existe um conjunto ou lista de candidatos como, por exemplo, os objetos que podem ser selecionados para preencher a mochila.

- Na medida em que o algoritmo procede, dois outros conjuntos são acumulados. Um contém candidatos que foram considerados, e o outro contém candidatos que foram considerados e rejeitados.

- Existe uma função que verifica se um conjunto particular de candidatos produz uma *solução* para o problema, sem considerar questões de **otimalidade** naquele momemento. Por exemplo, os objetos selecionados preenchem a mochila?

- Uma segunda função verifica se um conjunto de candidatos é *viável*, isto é, se é possível completar o conjunto adicionando mais candidatos de tal forma que pelo menos uma solução possa ser obtida. Aqui também não existe preocupação com a otimalidade.

- Uma outra função, chamada *função de seleção*, indica a qualquer momento quais dos candidatos restantes que não foram nem escolhidos nem rejeitados é o mais promissor.

- Finalmente, uma *função objetivo* fornece o valor da solução encontrada, como a utilidade alcançada com os objetos selecionados. Ao contrário das três funções mencionadas acima, a função objetivo não aparece explicitamente no algoritmo Guloso.

Um algoritmo Guloso genérico é apresentado no Algoritmo 11. Inicialmente, o conjunto S de candidatos escolhidos está vazio. A seguir, a cada passo, o melhor candidato restante ainda não tentado é considerado, sendo o critério de escolha ditado pela função de seleção. Se o conjunto aumentado de candidatos se torna inviável, o candidato corrente é rejeitado e nunca mais considerado. Entretanto, se o conjunto aumentado é viável então o candidato é adicionado ao conjunto S de candidatos escolhidos, onde permanece a partir de então. Cada vez que o conjunto S é aumentado ao receber um novo candidato, é necessário verificar se S constitui uma solução. Quando um algoritmo Guloso funciona corretamente, a primeira solução encontrada dessa maneira é sempre ótima. Entretanto para o problema da mochila em questão, não é possível afirmar se a solução obtida é ótima ou não.

A função de seleção é geralmente relacionada com a função objetivo. Se o objetivo é maximizar o ganho, então provavelmente será escolhido qualquer um dos candidatos restantes que proporcione o maior ganho individual. Se o objetivo é minimizar o custo, então será escolhido o candidato restante de menor custo, e assim por diante. Além disso, o algoritmo nunca muda de idéia. Uma vez que um candidato é escolhido e adicionado à solução, ele lá permanece para sempre. Da mesma maneira, uma vez que um candidato é excluído do conjunto solução, ele nunca mais é reconsiderado.

Nessa abordagem, não foi despendido muito esforço, pois não é conhecido um algoritmo Guloso que obtenha uma solução ótima, sempre. No entanto, são apresentados três algoritmos Gulosos, ou três funções escolha (*select*) diferentes. Mais precisamente, é apresentado um único algoritmo, mas os objetos são ordenados segundo três ordens, visando selecionar os objetos de três formas diferentes, quais sejam:

Algorítmo 11 *Greedy* - Algoritmo Guloso genérico.

procedure $Greedy(C : set)set$;
$\{C : setofcandidates\}$
begin
$S \leftarrow \emptyset$
while $(C \neq \emptyset)and(notsolution(S)$ **do**
  $x \leftarrow select(C)$
  $C \leftarrow C - x$
  if $feasible(S + x)$ **then**
    $S \leftarrow S + x$
  end if
end while
if $solution(S)$ **then**
  return (S)
else
  return $(\emptyset)$ {There aren't solution}
end if
end

- ordenação por utilidade - os objetos são ordenados de forma descendente de acordo com a utilidade de cada objeto. Após essa ordenação os objetos são tais que: $p_i \geq p_{i+1}$, $1 \leq i \leq n-1$. Essa ordenação visa fornecer ao conjunto solução o objeto com a maior utilidade possível em cada iteração, tentando, dessa forma, maximizar a utilidade, ou seja, a função objetivo.

- ordenação por peso - os objetos são ordenados de forma ascendente de acordo com o peso de cada objeto. Após essa ordenação os objetos são tais que: $w_i \leq w_{i+1}$, $1 \leq i \leq n-1$. Essa ordenação visa fornecer ao conjunto solução o objeto com o menor peso possível em cada iteração, tentando, dessa forma, inserir o maior número de objetos na mochila, permitindo maximizar a utilidade, ou seja, a função objetivo.

- ordenação pela razão entre a utilidade e o peso - os objetos são ordenados de forma descendente de acordo com a razão entre a utilidade e o peso de cada objeto. Após essa ordenação os objetos são tais que: $p_i/w_i \geq p_{i+1}/w_{i+1}$, $1 \leq i \leq n-1$. Essa ordenação visa fornecer ao conjunto solução o objeto com a maior utilidade por unidade de peso possível em cada iteração.

Então, um algoritmo Guloso para o problema da mochila para essas três abordagens deve simplesmente mudar a forma de ordenação do conjunto de objetos. No Algoritmo 12 é apresentado uma forma Gulosa de se resolver o problema da mochila.

Uma implementação não tão fiel ao Algoritmo 12 é apresentado no Programa 13. Essa implementação, no entanto segue o paradigma de uma abordagem Gulosa, simplesmente não explicita as funções seleção, objetivo e *feasible*. Essas funções estão espalhados dentro da própria implementação do método *GKnapSack()*.

O parâmetro, *opt*, que esse método, $GKnapSack()$, recebe indica o tipo de chave a ser gerada. Os métodos $GenerateKeyByProfit()$, $GenerateKeyByWeight()$ e $GenerateKeyByRatio()$ geram as chaves (*Key*, campo da estrutura *Object*) na estrutura *m_pData* de acordo com a utilidade, P, o peso W e a razão entre a utilidade

Algorítmo 12 *GKnapSack* - Algoritmo Guloso para o problema da mochila.

procedure $GKnapSack(n, M, W, P, KeyKind, Asc/Desc)$
$\{n, thenumberofobjects\}$
$\{M, theknapsacksize\}$
$\{w_i \in W$ and $p_i inP$ are the weight and profit of i object, respectively$\}$
$\{x_i \in X$ say if i object belongs to (1) or don't (0) to solution set $X\}$
$\{KeyKind$ is the way by sort: $\{Profit, Weight, Profit/Weight\}\}$
$\{Asc/Desc$ define the sequence in which data must be sorted: $Ascendent/Descendent\}$
begin
$GenerateKeyBy((W, P), KeyKind)$
$SortByKey((W, P), Asc/Desc)$
$X \leftarrow \emptyset$
while $((W, P) \neq \emptyset) and (notsolution(X)$ **do**
  $x \leftarrow select((W, P))$
  $(W, P) \leftarrow (W, P) - x$
  if $feasible(X + x)$ **then**
    $X \leftarrow X + x$
  end if
end while
if $solution(X)$ **then**
  return (X)
else
  return $(\emptyset)$ {There aren't solution}
end if
end

e o peso, respectivamente. O método de ordenação utilizado foi o *Heapsort*. A função *Heapsort* utilizada encontra-se nos anexos, no arquivo heapsort.cpp. Essa função recebe como parâmetros de entrada, o vetor a ser ordenado, no caso desse trabalho um vetor de *Object*'s, o tamanho do vetor, *m_ in*, e uma variável booleana que indica se os valores devem ser ordenados de forma ascendente (1) ou descendente (0).

```
void CKnapSack::GKnapSack(int opt)
{
  int i;
  int cw,cp;

  StartProcess();
  CreateDataSolution();

  switch(opt)
  {
  case 1:GenerateKeyByProfit ();Heapsort(m_pData,&m_in,0);break;
  case 2:GenerateKeyByWeight ();Heapsort(m_pData,&m_in,1);break;
  case 3:GenerateKeyByRatio  ();Heapsort(m_pData,&m_in,0);break;
  }

  cw = 0;
  cp = 0;
  for(i = 1 ; i<=m_in ; i++) /* generate S^i */
  {
    if (g_bOverflow) break;

    if (cw + m_pData[i].W<=m_iM)
    {
      cw += m_pData[i].W;
      cp += m_pData[i].P;
      m_piX[i] = 1;
    }
    else
      m_piX[i] = 0;
  }
  m_ifw = cw;
  m_ifp = cp;

  /* off oveflow alarm */
  alarm(0);

  if (!g_bOverflow)
  {
    VerifyResult();
    m_bSolution = TRUE;
  }
  /* P(1..n), W(1..n), X(1..n) + k */
  m_iSpace = 3*m_in;
  FinishTime();
}
```

Programa 13: Implementação do algoritmo Guloso para o problema da mochila.

Seja o seguinte exemplo, para ilustrar o funcionamento dos algoritmos Gulosos: o número de objetos $n = 4$, o tamanho da mochila $M = 6$, e $(W, P) = \{(1, 2), (2, 3), (4, 7), (5, 4)\}$.

Seja o conjunto de objetos, ordenados pela **utilidade**, $(W, P) =$

$\{(4,7),(5,4),(2,3),(1,2)\}$, executando o algoritmo Guloso, tem-se: $X = \{(4,7),(2,3)\}$, $fw = 6$ e $fp = 10$, que é equivalente a solução ótima.

Seja o conjunto de objetos, ordenados pelo **peso**, $(W,P) = \{(1,2),(2,3),(4,7),(5,4)\}$, executando o algoritmo Guloso, tem-se: $X = \{(1,2),(2,3)\}$, $fw = 3$ e $fp = 5$, que é uma solução sub-ótima para o problema.

Seja o conjunto de objetos, ordenados pela **razão entre a utilidade e o peso**, $(W,P) = \{(1,2),(4,7),(2,3),(5,4)\}$, executando o algoritmo Guloso, tem-se: $X = \{(1,2),(4,7)\}$, $fw = 5$ e $fp = 9$, que é uma solução sub-ótima para o problema.

Para a instância exemplo exposta, a abordagem Gulosa que utiliza ordenação pela utilidade conseguiu uma solução ótima. Isso foi casual, uma particularidade da instância exposta. Embora não seja possível afirmar que sempre essas abordagens produzirão soluções ótima, a abordagem que utiliza os objetos ordenados segundo a razão entre utilidade e peso produz soluções mais próximas à ótima, ou seja, que tem o menor erro médio. Os resultados expostos na Seção 6 confirmarão isto.

Sem levar em conta o tempo levado pela ordenação do conjunto de objetos, (*Heapsort* - $O(n \log n)$ - estável), cada uma dessas três estratégias tem complexidade de tempo de somente $O(n)$. Em relação a complexidade de espaço o mesmo pode ser dito. Somente um vetor, de tamanho n, é necessário para armazenar a solução e então a complexidade de espaço é também $O(n)$.

5.4 Resumo: Análises de Complexidade

Nessa subseção são sumarizadas as ordens de complexidade de tempo e de espaço, das abordagens apresentadas.

- Complexidade de Tempo
  - *Backtracking* - $O(2^n)$
  - Programação Dinâmica - $O(min\{2^n, n\sum_{i=1}^{n} p_i, nM\})$
  - *Greedy* - $O(n)$
- Complexidade de Espaço
  - *Backtracking* - $O(n)$
  - Programação Dinâmica - $O(min\{2^n, n\sum_{i=1}^{n} p_i, nM\})$
  - *Greedy* - $O(n)$

6 Experimentos

Nessa seção são apresentados os experimentos realizados para este trabalho. Os conjuntos de instânicas que foram utilizados são os descritos na Seção 4.

As três variáveis, *m_ bOptimal*, *m_ bSolution*, *m_ bOverflow*, de controle de estado que foram construídas na classe *CKnapSack* permitem determinar o estado da solução da instância retornada pelo sistema. A princípio, ao iniciar a execução, as três são ajustadas para *falso*. *m_ bSolution* só se torna *verdadeiro* caso uma solução seja encontrada. *m_ bOptimal* só se torna *verdadeiro* caso as abordagens *Backtracking* e *Dynamic Programming* são executadas por inteiro, pois elas garantem uma solução ótima. *m_ bOverflow* é ajustada para *verdadeiro* caso o tempo de execução da instância, pré-definido em 5 segundos, seja atingido.

Através da saída destas três varíaveis foi possível criar classificações para as soluções obtidas, ou seja:

- OTM - Solução Ótima (*m_ bOptimal* = *verdadeiro*, *m_ bSolution* = *verdadeiro*, *m_ bOverflow* = *falso*), retornada somente pelas abordagens *Backtracking* e Programação Dinâmica;
- FAL - Falha (*m_ bSolution* = *verdadeiro*, *m_ bOptimal* = *don't care*, *m_ bOverflow* = *don't care*), nenhuma instância apresentou essa situação;
- OTWO - Solução Ótima não esperada sem estouro de tempo (*overflow*) (*m_ bSolution* = *verdadeiro*, *m_ bOptimal* = *falso*, *m_ bOverflow* = *falso*), retornada pela abordagem Gulosa;
- OTO - Solução Ótima não esperada com estouro de tempo (*overflow*) (*m_ bSolution* = *verdadeiro*, *m_ bOptimal* = *falso*, *m_ bOverflow* = *verdadeiro*, retornada pela abordagem *Backtracking*;
- SOW - Solução Sub-Ótima não esperada sem estouro de tempo (*overflow*) (*m_ bSolution* = *verdadeiro*, *m_ bOptimal* = *falso*, *m_ bOverflow* = *falso*, retornada somente pelas abordagens Gulosas (nas instâncias testadas);
- SOO - Solução Sub-Ótima não esperada com estouro de tempo (*overflow*) (*m_ bSolution* = *verdadeiro*, *m_ bOptimal* = *falso*, *m_ bOverflow* = *verdadeiro*, nenhuma instância apresentou essa situação.

Essa análise em relação a solução ótima foi possível, pois contáva-se com a solução ótima para todas as instâncias. Essas soluções foram obtidas com o algoritmo de Programação Dinâmica que conseguia retornar uma solução em tempo eficiente (menos de 1 segundo na média). Qualquer solução retornada pela abordagem de Programação Dinâmica é ótima, pois esta é uma propriedade do algoritmo implementado. A questão é saber se ela devolverá uma solução em tempo eficiente? Para as instâncias do trabalho, o algoritmo implementado sempre conseguiu isso.

Os experimentos realizados sobre as 2000 instânicas, estão reportados nas Tabelas de 4 até 19. As colunas referentes as classificações FAL e SOO foram omitidas das tabelas, pois nenhuma instância dos experimentos foram classificadas nelas. Cada coluna, que representa um tipo de classificação, dessas tabelas contém o número de instâncias *num*, que foram classifadas para o tipo, o tempo ($Time$) médio em milisegundos que as instâncias levaram para ser executadas e o espaço (*Space*) médio ocupado em bytes. As tabelas com tamanho de mochila em 40% foram omitidas, por um *bug*/limitação do latex com relação ao número de *float*'s no documento. Entretanto, a eliminação dessas 4 tabelas (uma para cada tipo de conjunto de instâncias) não compromete a análise que se fará aqui.

Nas próximas subseções, serão realizadas análises em separado sobre os resultados dos experimentos. Serão construídos gráficos para representar/mostrar o comportamento das abordagens com relação a variação do tamanho percentual da mochila PM, o número de objetos n e o tipo de conjunto de instâncias.

6.1 *Backtracking*

Analisando os dados apresentados nas tabelas anexadas (Tabelas de 4 até 19) pode-se extrair algumas conclusões em relação aos algoritmos $BKnapSack1$ e $BKnapSack2$:

- Os algoritmos *Backtracking* foram muito eficientes (menos de 1 de milisegundo) para resolver todas as instâncias dos conjuntos 1, 2 e 4;

- Os algoritmos *Backtracking* foram muito eficientes (menos de 5 de milisegundo) para resolver todas as instâncias dos conjuntos 3, com n inferior a 100 objetos;

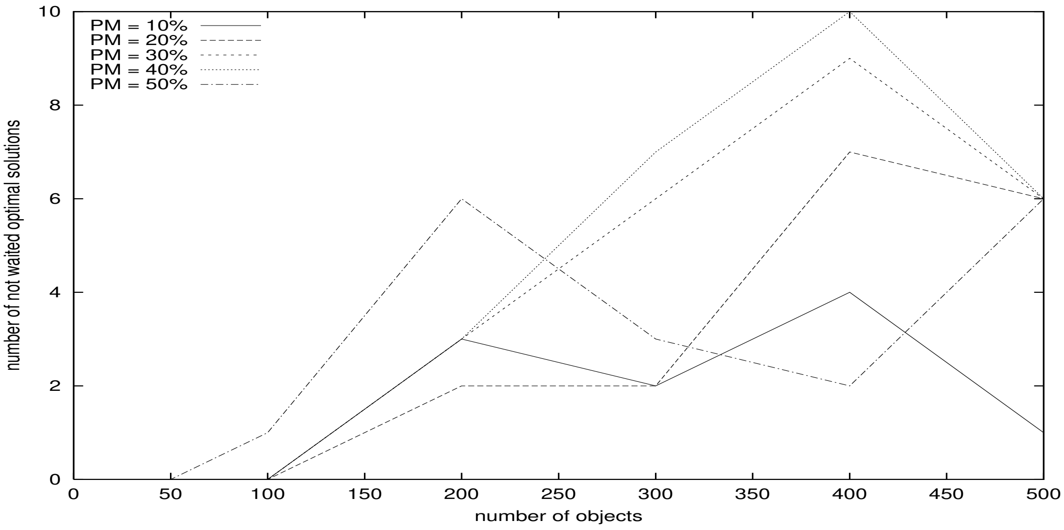
- no conjunto de instâncias do tipo 3, o valor $n = 100$ objetos foi o ponto no qual começaram a ocorrer estouros de tempo de processamento.

- Infelizmente, para as instâncias do conjunto tipo 3, com n superior a 100 objetos, os algoritmos estouraram o tempo limite de execução (5 segundos), acredita-se que eles não foram capazes de podar muitos galhos. Todavia, o mais interessante aconteceu, após esses cinco segundos, em todas as instâncias que ocorreram estouro, a solução encontrada pelo algoritmo, embora não possa ser dita ótima, apresentava o valor da função objetivo comparável ao valor da função objetivo de uma solução ótima. Novamente, ressalta-se que essas soluções eram fornecidas pelo algoritmo de Programação Dinâmica, que embora não fosse tão rápido quando o *Backtracking* em instâncias ditas fáceis, conseguia a solução em tempo inferior a 1 segundo para todas as instâncias com $n \leq 500$;

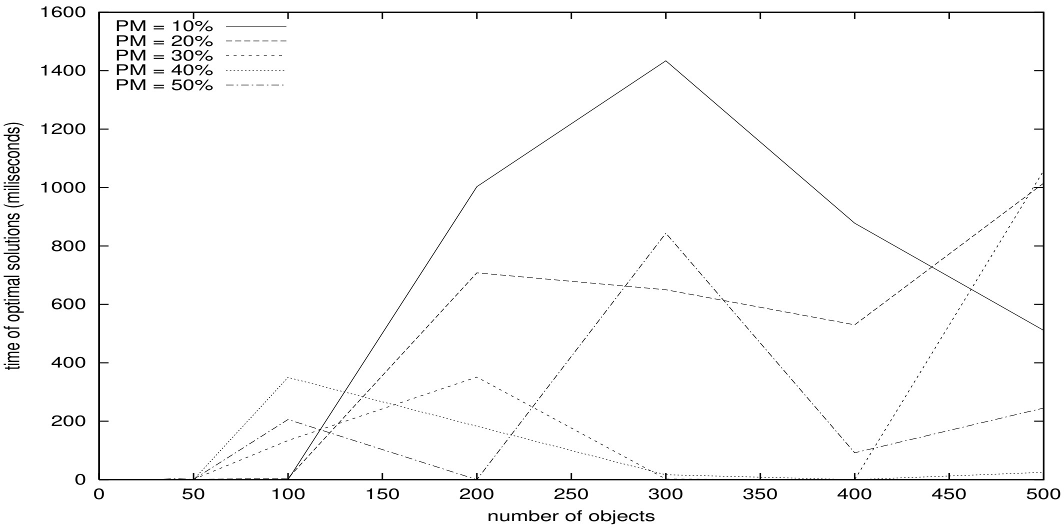
- Não foram notadas diferenças significativas entre os tempos de execução dos algoritmos $BKnapSack1$ e $BKnapSack2$. A otimização sugerida em [1] não surgiu muito efeito, isso talvez seja explicado pela evolução dos compiladores e também porque a simplificação foi feita sobre o número de computações sobre uma iteração e não foi feito nenhuma redução de estado a serem visitados.

Considera-se conveniente, explorar o espaço onde o algoritmo *Backtracking* não foi tão eficiente. Portanto, mostrar-se-á alguns gráficos relacionados ao conjunto de instâncias do tipo 3 e descartar-se-á os resultados obtidos pelo algoritmo $BKnapSack1$ e considerar-se-á somente os de $BKnapSack2$, pois ele são muito similares.

Na Figura 4 são apresentadas dois gráficos relacionados com os conjuntos de instâncias do tipo 3. Na Figura 4(a) apresenta-se um gráfico do número de objetos contra o número de soluções ótimas não esperadas (com estouro de tempo) pelo



(a)



(b)

Figura 4: Gráficos sobre os resultados dos experimentos da abordagem *Backtracking* para o conjunto de instâncias tipo 3 ($w_i = [1, 100]$ e $p_i = w_i + 10$).

algoritmo $BKnapSack2$. Já na Figura 4(b) apresenta-se um gráfico do tempo decorrido para o algoritmo $BKnapSack2$ encontrar as soluções ótimas, já que para as soluções ótimas não esperadas com estouro o tempo delas é de 5 segundos.

Embora a idéia de plotar esses dois gráficos fosse a de clarificar o comportamento do algoritmo $BKnapSack2$ nessas situações, parece que a situação foi contrária. É díficil extrair alguma informação dos gráficos. Uma análise e reflexão profunda leva a seguinte conclusão: como a natureza das instâncias desses objetos é aleatória e o funcionamento do mesmo é muito dependente dos valores de pesos e de utilidades, tem-se que em algumas instâncias com tamanho de 400 objetos o algoritmo consiga solução em menos de 10 milisegundos, já para algumas instâncias de 100 objetos ocorre o tal *overflow*. Em suma, o tempo de execução desse algoritmo está intimamente ligado a natureza dos dados da instância.

A demanda por espaço desses algoritmos é linear ($O(n)$), apesar de varrer um espaço exponencial ($O(2^n)$). Somente o melhor caminho (solução) percorrido e o caminho atual são armazenados. O melhor caminho ocupa um vetor de tamanho do número de objetos, já o atual caminho percorrido pode ocupar no máximo o mesmo tamanho. Essas medida foram facilmente computadas e podem ser verificadas pelas tabelas em anexo.

6.2 Programação Dinâmica

Analisando os dados apresentados nas tabelas anexadas (Tabelas de 4 até 19) pode-se extrair algumas conclusões em relação aos algoritmos $DKnapSack1$ e $DKnapSack2$:

- Um comportamento mais estável dos tempos de execução em relação aos algoritmos *Backtracking*;

- Um comportamento pseudo-exponencial em função do número de objetos n tanto para o espaço quanto para o tempo;

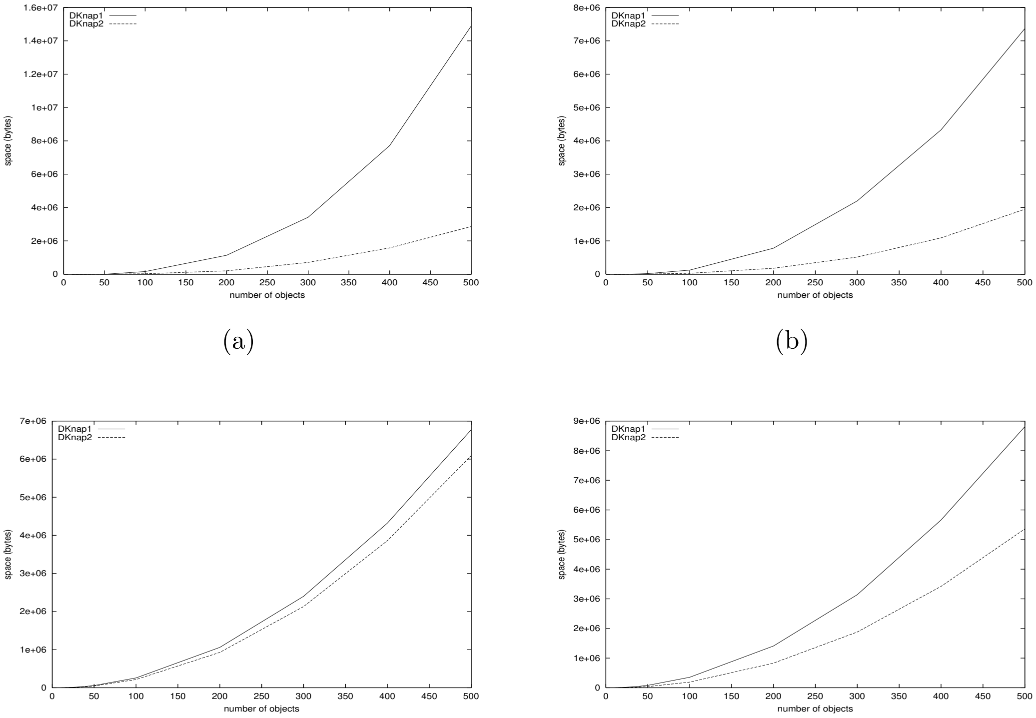
- Para os conjuntos de instâncias de tipo 1, 2 e 4 os algoritmos de Programação Dinâmica tiveram desempenho inferior aos algoritmos de *Backtracking* em relação ao tempo. No entanto, para o conjunto de instâncias de tipo 3 ele se manteve com desempenho similar aos apresentados nos conjuntos de instância 1, 2 e 4. Isto permite estabelecer uma analogia entre os algoritmos de ordenação *Heapsort* e *Quicksort*. A abordagem Programação Dinâmica está para o algoritmo *Heapsort*, assim como a abordagem *Backtracking* está para o algoritmo de ordenação *Quicksort*. Ou seja, a abordagem Programação Dinâmica tem um comportamente mais estável, mantendo um desempenho do pior caso próximo ao caso médio, enquanto que a abordagem *Backtracking* é muito eficiente no caso médio e seu desempenho despenca no pior caso.

- o algoritmo $DKnapSack2$ é superior ao algoritmo $DKnapSack1$ nos conjuntos de instâncias 1, 2 e 4. Já para o conjunto de instâncias 3 o desempenho dos algoritmos é similar. Isto vale tanto para a complexidade de tempo e de espaço e pode ser confirmado, analisando os gráficos das Figuras 5 e 6.

Para ilustrar o comportamento estável apresentado pelos algoritmo de Programação Dinâmica são plotados gráficos para cada tipo de conjunto de instância e com tamanho percentual da mochila de $PM = 50\%$. Nesses conjuntos de instâncias com $PM = 50\%$ houve uma maior demanda por espaço e conseqüentemente por tempo de processamento.

O espaço utilizado pelos algoritmos de Programação Dinâmica é computado como sendo $\sum_{i=1}^{n-1} |S^i|$. O espaço ocupado pelas soluções com $PM = 50\%$ pelos conjuntos de instâncias 1, 2, 3 e 4 são apresentados, em forma gráfica, nas Figuras 5(a), 5(b), 5(c) e 5(d), respectivamente. Para a solução dos maiores problemas/instâncias foi utilizado 14 *Megabytes* de memória *RAM* na média.

Comparando-se os espaços utilizados entre os algoritmos $DKnapSack1$ e $DKnapSack2$ percebe-se que a otimização proposta para o algoritmo $DKnapSack2$



(a) (b)

(c) (d)

Figura 5: Gráficos sobre os resultados dos experimentos, referentes ao espaço utilizado, da abordagem de Programação Dinâmica para um tamanho percentual da mochila de $PM = 50\%$.

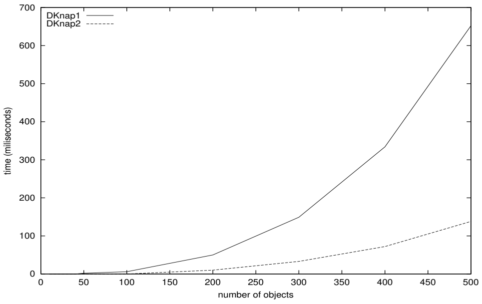
é muito significativa para os conjuntos de instâncias do tipo 1, 2 e menos pronunciada para o conjunto de instâncias 4. Já para o conjunto de instância 3, os algoritmos $DKnapSack1$ e $DKnapSack2$ tem uso de memória (espaço) muito próximos.

Os tempos de execução dos algoritmos de Programação Dinâmica são proporcionais ao espaço utilizado conforme pode ser notado, comparando-se visualmente as Figuras 5 e 6. Pode-se também notar a natureza pseudo-exponencial de execução da abordagem de Programação Dinâmica. As demais análises realizadas para o uso de memória (espaço) podem ser estendidas para o tempo.

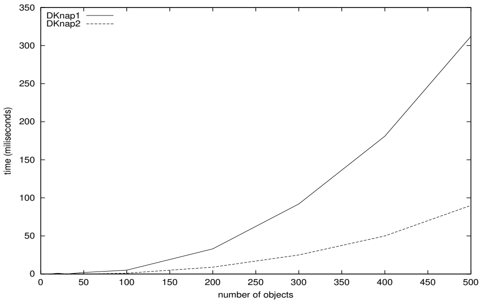
6.3 Gulosa

O quantidade de experimentos realizados foi grande. Com o intuito de mostrar o comportamento geral dos algoritmos Gulosos em relação a variação do número de objetos n e o erro relativo da solução obtida com a solução ótima, gerou-se quatro gráficos, um para cada tipo de conjunto de instâncias, conforme pode ser visto na Figura 7. Os comportamentos das três abordagens Gulosas são apresentados para os tipos de conjunto 1, 2, 3 e 4 nas Figuras 7(a), 7(b), 7(c) e 7(d), respectivamente. Observando essas quatro figuras, pode-se tirar algumas conclusões:

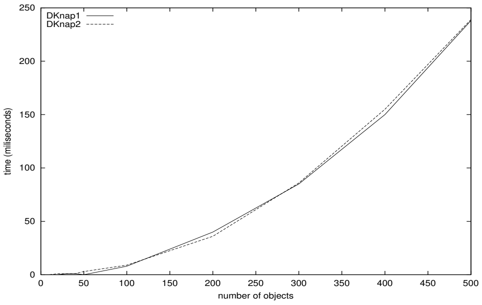
- A abordagem Gulosa com ordenação por utilidade *profit*, tem menor erro relativo para as instâncias muito pequenas ($n = 10$);



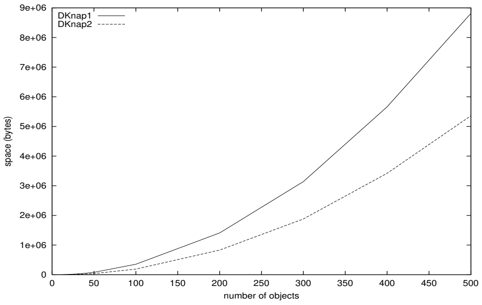
(a)



(b)



(c)



(d)

Figura 6: Gráficos sobre os resultados dos experimentos, referentes ao tempo de processamento, da abordagem de Programação Dinâmica para um tamanho percentual da mochila de $PM = 50\%$.

- De uma forma geral, a abordagem Gulosa com ordenação pela razão entre utilidade e peso apresenta o menor erro relativo em todos os casos, excluindo, lógicamente, a situação onde $n = 10$;

- Para o tipo de conjunto de instânicas 3 o erro relativo entre as abordagens de ordenação por peso e pela razão entre utilidade e peso são coincidentes. Isto é justificado pela forma que o conjunto de instâncias 3 foi gerado. Nesse conjunto a utilidade é simplesmente o peso adicionado de uma constante (10), o que permite concluir que essas curvas devem ser mesmo coincidentes, pois os objetos quando ordenados tanto pela razão entre utilidade e peso em ordem descendente quanto pelo peso em ordem ascendente, possuem a mesma ordem.

Seria interessante uma inspeção sobre as 18 tabelas (Tabelas de 4 até 19) expostas na Seção 9. Através delas poderia ser verificado que em algumas situações existem soluções ótimas não esperadas, produzidas pelos algoritmos Gulosos. Embora, mais uma vez, isso não é garantido - soluções ótimas, elas são casuais - para essa abordagem, neste problema.

Todas essas análises são restritas a essas 2000 instâncias e não são verdades absolutas, pois as conclusões são altamente dependente dos valores da utilidade e peso das instâncias.

Sobre os tempos médios dos experimentos para os algoritmos gulosos foi impossível ter uma medida precisa, no entanto, verificou-se que para todos os tamanhos

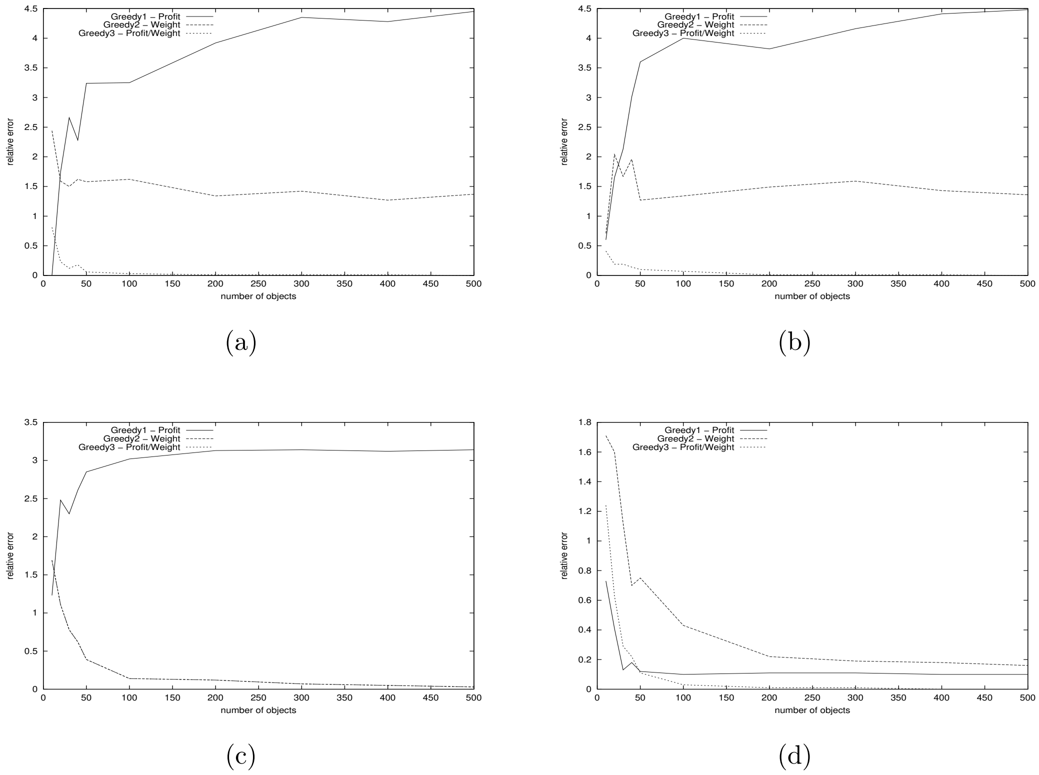


Figura 7: Gráficos sobre os resultados dos experimentos da abordagem Gulosa *Greedy* para $PM = 10\%$.

de n e para todos os tipos de conjuntos de instância e para todos os tamanho percentuais de mochila, ou seja, para todas as instâncias, obteve-se tempo bem inferior a 10 milisegundos (medida mínima de tempo da arquitetura utilizada). Sabe-se que este valor é inferior a 10 milisegundos, comparando-se com o tempo total para execução de todas essas instâncias para cada uma das abordagens, ou seja, o tempo tomado para executar as 2000 instâncias é de aproximadamente 1 segundo (Pentium 4, dual, com 1 GygaByte de RAM - *jade.dcc.ufmg.br*). Isto já era esperado pois este algoritmo tem complexidade de tempo linear $O(n)$. Para o espaço, análise similar é valida, visto que o algoritmo tem complexidade de espaço linear $O(n)$, este fato pode ser confirmado pelas tabelas anexadas (Tabelas de 4 até 19). Entretanto, para a demanda de espaço foi possível calcular o espaço utilizado por cada instância facilmente.

6.4 Resumo: Maiores amostras resolvidas

- *Backtracking* - para todas as instâncias dos conjuntos de instâncias 1, 2 e 4, os algoritmos conseguiram solução em menos de 1 segundo. Já para o conjunto de instância 3, a partir de 100 objetos os algoritmos se tornaram ineficientes para resolver as instâncias, estourando o tempo limite ajustado em 5 segundos. Apesar de estourarem o tempo limite, as soluções produzidas até o instante do estouro produziam uma função objetivo idêntica as da solução ótima, ou seja, as soluções podem ser ditas ótimas não esperadas;

- Programação Dinâmica - para todas as instâncias com 500 objetos os algoritmos propostos conseguiram solução ótima em menos de 1 segundo. Para as instâncias com tamanho de 500 objetos o espaço necessário variou de 1 MegaBytes a 20 MegaBytes;

- *Greedy* - todas as instânicas foram resolvidas em um tempo bem infeior a 10 milisegundos e o erro relativo à solução ótima ficou abaixo de 5% para todas as instâncias.

Todas essas análises de tempo estão levando em conta o tempo de ordenação dos objetos segundo algum critério pré-estabelecido. O algoritmo utilizado foi o *Heapsort*[5], que é $O(n \log n)$ para qualquer caso. Esse tempo de ordenação é constante para instâncias com mesmo n e muito pequeno. Então, esse tempo foi desprezado em todas as análises efetuadas.

7 Sistema

Nessa seção são apresentados os módulos que foram utilizados para automatizar todo o processo de realização desse trabalho. Esses módulos estão encapsulados em uma classe chamada *CSystem* que está listada no Programa 14. Nessa classe existem quatros métodos que implementam os quatro módulos desse sistema.

```
class CSystem
{
private:
  int     m_iargc;
  char ** m_ppcargv;

  int m_ids    ,m_idf;
  int m_iInsts,m_iInstf;
  int m_iMs    ,m_iMf;
  int m_ins    ,m_inf;

  int m_iOpt1,m_iOpt2; /* selection for approach's */

  char sExt[4][4][6];

  /* data for resume */
  int    kind[4][10][5][100]; /* kind of solution 0,1,2,3,4,5 */
  int    time[4][10][5][100]; /* taken time  for instance solution */
  int    spc [4][10][5][100]; /* taken space for intance solution */
  int    num [4][10][5][6];   /* number of kind solution */
  int    stem[4][10][5][6];   /* sum of solutions time of kind */
  int    sesp[4][10][5][6];   /* sum of solutions space of kind */
  double ssot[4][10][5][6];   /* sum of suboptimal solution percentual of kind */

  /* temporary data used in some modules */
  char sPathData[BUFSIZ],sFileOut[BUFSIZ],sTmp[BUFSIZ];

  int    dt,n,inst,b; /* counter */
  double dPM; /* counter */
  FILE   *pfOut; /* Output file results */

  CKnapSack* m_poKnapSackApproach;
  CKnapSack* m_poKnapSackOptimal;

private:
  void CompareSolutions();

public:
  CSystem();
  ~CSystem();

  void SetArgs(int argc,char** argv);

  int ModuleFile      ();
  int ModuleBatch     ();
  int ModuleGenerator ();
  int ModuleResume    ();
};
```

Programa 14: classe *CSystem* que implementa o Sistema

Quando o programa principal (*knapsack*) é executado sem parâmetros, os quatro módulos que estão disponíveis para execução e seus respectivos parâmetros são exibidos:

```
./knapsack
error, not enough parameters
usage: ./knapsack file      option1 option2 filedata
usage: ./knapsack batch     option1 option2 pathdata ds-df ns-nf
                            Ms-Mf insts-instf
usage: ./knapsack generator pathdata ds-df ns-nf Ms-Mf insts-instf
usage: ./knapsack resume    pathdata ds-df Ms-Mf
```

Os quatro módulos apresentados pelo sistema *file*, *batch*, *generator* e *resume* possibilitam avaliar uma única instância por vez, um conjunto de instâncias, gerar instâncias e fazer um resumo relativo aos experimentos realizados, respectivamente. Nas próximas subseções cada módulo será detalhado.

7.1 Módulo *file*

O módulo *File*, implementado pelo Programa 15, permite que uma instância seja submetida aos algoritmos para solução do problema. Esse módulo tem a seguinte sintaxe:

```
./knapsack file      option1 option2 filedata
```

onde *file* indica o módulo a ser ativado, *option1* e *option2* correspondem aos algoritmos a serem executados de acordo com a Tabela 1 e *filedata* o nome do arquivo a ser executado.

Tabela 1: Referências de *option1* e *option2* em relação aos algoritmos a serem executados.

Algoritmo	*option1*	*option2*
$BKnapSack1()$	1	1
$BKnapSack2()$	1	2
$DKnapSack1()$	2	1
$DKnapSack2()$	2	2
$GKnapSack(1)$	3	1
$GKnapSack(2)$	3	2
$GKnapSack(3)$	3	3

```
int CSystem::ModuleFile        ()
{
  int m_iOpt1,m_iOpt2;
  char sFileIn[BUFSIZ];
```

```
  m_poKnapSackApproach = new CKnapSack();

  m_iOpt1 = atoi(m_ppcargv[2]);
  m_iOpt2 = atoi(m_ppcargv[3]);
  strcpy(sFileIn,m_ppcargv[4]);

  if ( (m_poKnapSackApproach->FileLoadInstance(sFileIn) == 0) )
  {
    fprintf(stdout,"error loading instance!\n");
    return 0;
  }

  /* approaches */
  switch(m_iOpt1)
  {
  case 1:
    switch(m_iOpt2)
    {
    case 1:m_poKnapSackApproach->BKnapSack1();break;
    case 2:m_poKnapSackApproach->BKnapSack2();break;
    }
    break;
  case 2:
    switch(m_iOpt2)
    {
    case 1:m_poKnapSackApproach->DKnapSack1();break;
    case 2:m_poKnapSackApproach->DKnapSack2();break;
    }
    break;
  case 3:m_poKnapSackApproach->GKnapSack(m_iOpt2);
  }

  m_poKnapSackApproach->ShowDataSolution();

  delete m_poKnapSackApproach;

  return 0; /* ok */
}
```

Programa 15: Módulo *file* implementado pelo método *ModuleFile*() da classe *CSystem*

7.2 Módulo *batch*

O módulo *batch*, implementado pelo Programa 16, permite que um conjunto de instâncias seja submetida aos algoritmos para solução do problema. Esse módulo tem a seguinte sintaxe:

```
./knapsack batch     option1 option2 pathdata ds-df ns-nf
                     Ms-Mf insts-instf
```

onde *batch* indica o módulo a ser ativado, *option1* e *option2* correspondem aos algoritmos a serem executados de acordo com a Tabela 1, *pathdata* indica o caminho onde as instâncias estão armazenadas, ds e df indicam o tipo de conjunto de instâncias, d, inicial e final, respectivamente, ns e nf indicam o número de objetos, n, inicial e final, respectivamamante, dentro do conjunto $n = \{10, 20, 30, 40, 50, 100, 200, 300, 400, 500\}$, Ms e Mf indicam o tamanho percentual da mochila, PM, inicial e final, respectivamente, dentro do conjunto $PM = \{10\%, 20\%, 30\%, 40\%\}$ e $insts$ e $instf$ indicam o número da instância, $inst$ inicial e final, respectivamente. As instâncias que são submetidas a este módulo, devem obedecer aos padrões de nome de instâncias estabelecidos na Seção 4.

Tabela 2: Extensões utilizadas para os arquivos de cada algoritmo.

Algoritmo	Extensão
$BKnapSack1()$	.bkt1
$BKnapSack2()$	.bkt2
$DKnapSack1()$	.dyn1
$DKnapSack2()$	.dyn2
$GKnapSack(1)$	.grd1
$GKnapSack(2)$	.grd2
$GKnapSack(3)$	.grd3

```
int CSystem::ModuleBatch ()
{
  int t;
  /* batch */

  m_iOpt1 = atoi(m_ppcargv[2]);
  m_iOpt2 = atoi(m_ppcargv[3]);
  strcpy(sPathData,m_ppcargv[4]);
  parameter2(m_ppcargv[5],m_ids    ,m_idf    );
  parameter2(m_ppcargv[6],m_ins    ,m_inf    );
  parameter2(m_ppcargv[7],m_iMs    ,m_iMf    );
  parameter2(m_ppcargv[8],m_iInsts,m_iInstf);

  sprintf(sFileOut,"dset%1d-%1dn%03d-%03dM%03d-%03di%03d-%03d%s.out"
    ,m_ids,m_idf,m_ins,m_inf,m_iMs,m_iMf,m_iInsts,m_iInstf,sExt[m_iOpt1][m_iOpt2]);
  sprintf(sTmp,"%s%s",sPathData,sFileOut);
  pfOut = fopen(sTmp,"wt+");

  fprintf(pfOut,"dataset - %d - %d\n"                ,m_ids    ,m_idf    );
  fprintf(pfOut,"object's number - %d - %d\n"        ,m_ins    ,m_inf    );
  fprintf(pfOut,"knapsack size - %d%% - %d%%\n"      ,m_iMs    ,m_iMf    );
  fprintf(pfOut,"number of instances - %d - %d\n",m_iInsts,m_iInstf);

  CStackTime oStackTime;

  oStackTime.Start();

    for(dt=m_ids;dt<=m_idf;dt++)
  {
    b=(int)(log10(m_ins));
    for(n=m_ins;n<=m_inf;n+=(int)pow(10.,b))
    {
```

```
      for(dPM=m_iMs;dPM<=m_iMf;dPM+=10)
      {
        for(inst=m_iInsts;inst<=m_iInstf;inst++)
        {
          kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = 0;
          time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = 0;
          spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = 0;
        }
        for(t=0;t<6;t++)
        {
          num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t] = 0;
          stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t] = 0;
          sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t] = 0;
          ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t] = 0;
        }
      }
      if ( (n / pow(10,b)) == 5)
      {
        n = 0;
        b++;
      }
    }
  }

  for(dt=m_ids;dt<=m_idf;dt++)
  {
    for(dPM=m_iMs;dPM<=m_iMf;dPM+=10)
    {
      b=(int)(log10(m_ins));
      for(n=m_ins;n<=m_inf;n+=(int)pow(10.,b))
      {
        for(inst=m_iInsts;inst<=m_iInstf;inst++)
        {
          m_poKnapSackApproach = new CKnapSack();
          m_poKnapSackOptimal  = new CKnapSack();

          fprintf(pfOut,"dset%1dn%03dM%03di%03d ",dt,n,(int)dPM,inst);

          sprintf(sFileOut,"dset%1dn%03dM%03di%03d.dat",dt,n,(int)dPM,inst);
          sprintf(sTmp,"%s%s",sPathData,sFileOut);

          if (m_poKnapSackApproach->FileLoadInstance(sTmp) == 0)
          {
            fprintf(pfOut,"error loading instance!\n");
            continue;
          }

          if ( (m_poKnapSackApproach->Getn() != n) )
          {
            fprintf(pfOut,"wrong data (n) inside loaded instance!\n");
            continue;
          }

          m_poKnapSackOptimal->FileLoadInstance(sTmp);

          /* approaches */
          switch(m_iOpt1)
          {
```

```
      case 1:
        switch(m_iOpt2)
        {
        case 1:m_poKnapSackApproach->BKnapSack1();break;
        case 2:m_poKnapSackApproach->BKnapSack2();break;
        }
        break;
      case 2:
        switch(m_iOpt2)
        {
        case 1:m_poKnapSackApproach->DKnapSack1();break;
        case 2:m_poKnapSackApproach->DKnapSack2();break;
        }
        break;
      case 3:m_poKnapSackApproach->GKnapSack(m_iOpt2);
      }

      /* Load optimal solution */
      sprintf(sFileOut
        ,"dset%1dn%03dM%03di%03d.opt",dt,n,(int)dPM,inst);
      sprintf(sTmp,"%s%s",sPathData,sFileOut);
      if (m_poKnapSackOptimal->FileLoadSolution(sTmp) == 0)
      { }

      CompareSolutions();

      fprintf(pfOut,"time: %5dms, "
        ,m_poKnapSackApproach->GetTime());

      if (m_poKnapSackApproach->GetSpace() < 1000.)
        fprintf(pfOut," space: %6.2lf b "
          ,(double)m_poKnapSackApproach->GetSpace());
      else if (m_poKnapSackApproach->GetSpace() < 1000000.)
        fprintf(pfOut," space: %6.2lfKb "
          ,(double)m_poKnapSackApproach->GetSpace()/1000.);
      else if (m_poKnapSackApproach->GetSpace() < 1000000000.)
        fprintf(pfOut," space: %6.2lfMb "
          ,(double)m_poKnapSackApproach->GetSpace()/1000000.);
      else if (m_poKnapSackApproach->GetSpace() < 1000000000000.)
        fprintf(pfOut," space: %6.2lfGb "
          ,(double)m_poKnapSackApproach->GetSpace()/1000000000.);

      m_poKnapSackApproach->PrintState(pfOut);

      fflush(NULL);

      delete m_poKnapSackApproach;
      delete m_poKnapSackOptimal;
    }
    if ( (n / pow(10,b)) == 5)
    {
      n = 0;
      b++;
    }
  }

  FILE *pfRes;
  /* FileSave Resume */
```

```
      sprintf(sFileOut,"dset%1dM%03d%s.res"
        ,dt,(int)dPM,sExt[m_iOpt1][m_iOpt2]);
      sprintf(sTmp,"%s%s",sPathData,sFileOut);
      pfRes = fopen(sTmp,"wt+");
      /* header */
      fprintf(pfRes,"M= %3lf\n",dPM);
      fprintf(pfRes,"        |     OIM        |  |      OIWO     |
                             |      OIO       |  |          SOW          |\n");
      fprintf(pfRes," n    num Time  Space    num Time  Space
                             num Time  Space    num Time  Space    Error \n");
/*                    XXX  123 1234 123.12Kb  123 1234 123.12Kb
                      123 1234 123.12Kb  123 1234 123.12Kb 123.56%*/
      b=(int)(log10(m_ins));
      for(n=m_ins;n<=m_inf;n+=(int)pow(10.,b))
      {
        fprintf(pfRes,"%3d  ",n);
        for(t=0;t<6;t++)
        {
            if ( (t ==FAL) || (t ==SOO) )
                    continue;

          fprintf(pfRes,"%3d %4.1f  "
            ,num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]
            ,(num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]==0)?0.:
             (double)stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]/
                    num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]);
          double tesp  =
             (num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]==0)?0.:
             (double)sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]/
                    num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t];
          if (tesp < 1000.)
            fprintf(pfRes,"%6.2lf b ",tesp);
          else if (tesp < 1000000.)
            fprintf(pfRes,"%6.2lfKb ",tesp/1000.);
          else if (tesp < 1000000000.)
            fprintf(pfRes,"%6.2lfMb ",tesp/1000000.);
          else if (tesp < 1000000000000.)
            fprintf(pfRes,"%6.2lfGb ",tesp/1000000000.);

          if ( (t ==SOW) || (t ==SOO) )
            fprintf(pfRes,"%6.2lf%% "
              ,ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][t]);
          else
            fprintf(pfRes," ");
        }
        fprintf(pfRes,"\n");

        if ( (n / pow(10,b)) == 5)
        {
          n = 0;
          b++;
        }
      }
      fclose(pfRes);
    }
  }
  oStackTime.Finish();
```

```
    oStackTime.GetStringTime(sTmp);
    fprintf(pfOut,"Total Time: %s\n",sTmp);

    fclose(pfOut);

    return 0; /* ok */
}
```

Programa 16: Módulo *batch* implementado pelo método *ModuleBatch*() da classe *CSystem*

Para cada instância executada pelo módulo *batch* é gerado um arquivo com a solução obtida por ele, no seguinte formato:

$$dsetd\%1dn\%03dM\%03di\%03d\%s$$

onde os caracteres de controle %1d, %03d, %03d, %03d, e %s, são substituídos seqüencialmente por d, n, PM, $inst$ e ALG. Esse último obedece o padrão de extensões de arquivos da Tabela 2.

Essa solução do algoritmo é então comparada com a melhor solução encontrada pelos algoritmos até o momento (*.opt*). Caso ela não exista a solução encontrada pelo algoritmo é então salva com o seguinte formato:

$$dsetd\%1dn\%03dM\%03di\%03d.opt$$

O trabalho de comparar a melhor solução obtida até o momento (*.opt*) com a solução obtida pelo algoritmo corrente (ALG) é realizado pelo método *CompareSolutions*() da classe *CSystem* que está anexado ao trabalho. É através desse método que os dados estatísticos são gerados e posteriormente salvos como um arquivo de resumo.

Para cada conjunto de instâncias com o mesmo tipo de conjunto de instâncias d, mesma quantidade de objetos, n, e mesmo tamanho percentual da mochila PM é gerado um arquivo resumo com o seguinte padrão:

$$dsetd\%1dn\%03dM\%03d\%s.res$$

onde os caracteres de controle %1d, %03d, %03d e %s, são substituídos seqüencialmente por d, n, PM e ALG. Esse último obedece o padrão de extensões de arquivos da Tabela 2.

Um exemplo desses arquivos pode ser visto na Tabela 3. Cada um dos sete resumos existentes em cada tabelas (Tabelas 4 até 19) anexada ao trabalho são provenientes desses resumos.

Tabela 3: Exemplo de resumo gerado pelo módulo *batch*.

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	0.35%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.10%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.07%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.05%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.04%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.01%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.00%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

7.3 Módulo *generator*

O módulo *generator*, implementado pelo Programa 5, permite que um conjunto de instâncias seja gerado. Esse módulo tem a seguinte sintaxe:

```
./knapsack generator pathdata ds-df ns-nf Ms-Mf insts-instf
```

onde *generator* indica o módulo a ser ativado, *pathdata* indica o caminho onde as instâncias serão armazenadas, *ds* e *df* indicam o tipo de conjunto de instâncias, d, inicial e final, respectivamente, ns e nf indicam o número de objetos, n, inicial e final, respectivamante, dentro do conjunto $n = \{10, 20, 30, 40, 50, 100, 200, 300, 400, 500\}$, Ms e Mf indicam o tamanho percentual da mochila, PM, inicial e final, respectivamente, dentro do conjunto $PM = \{10\%, 20\%, 30\%, 40\%\}$ e $insts$ e $instf$ indicam o número da instância, $inst$, inicial e final, respectivamente. O formato do arquivo das instâncias geradas por esse módulo bem como o formato do nome das instâncias já foram discutidos na Seção 4 e sabe-se que esses arquivos servirão de entrada para o módulo *batch* de execução em lote.

7.4 Módulo *resume*

O módulo *resume*, implementado pelo Programa 17, permite que os resumos gerados pelo módulo *batch* para o mesmo tipo de conjuntos de instâncias d e mesmo tamanho percentual de mochilas PM sejam concatenados em um único arquivo de resumo. Esse módulo tem a seguinte sintaxe:

```
./knapsack resume    pathdata ds-df Ms-Mf
```

ds e *df* indicam o tipo de conjunto de instâncias, d, inicial e final, respectivamente e Ms e Mf indicam o tamanho percentual da mochila, PM, inicial e final, respectivamente.

O arquivo de resumo geral gerado tem o seguinte formato:

$$dsetd\%1dM\%03d.res$$

onde os caracteres de controle %1d, %03d, são substituídos seqüencialmente por d, PM.

```
int CSystem::ModuleResume    ()
{
  /* Resume */
  int opt_size[] = {0,2,2,3};
  int opt1,opt2;
  FILE *pfRes,*pfResXX;
  char sLine[BUFSIZ];

  strcpy(sPathData,m_ppcargv[2]);
  parameter2(m_ppcargv[3],m_ids,m_idf);
  parameter2(m_ppcargv[4],m_iMs,m_iMf);

  for(dt=m_ids;dt<=m_idf;dt++)
```

```
  {
    for(dPM=m_iMs;dPM<=m_iMf;dPM+=10)
    {
      sprintf(sFileOut,"dset%1dM%03d.res",dt,(int)dPM);
      sprintf(sTmp,"%s%s",sPathData,sFileOut);
      pfRes=fopen(sTmp,"wt+");
      fprintf(stdout,"%s\n",sTmp);

      for(opt1=1;opt1<=3;opt1++)
      {
        for(opt2=1;opt2<=opt_size[opt1];opt2++)
        {
          sprintf(sFileOut,"dset%1dM%03d%s.res"
            ,dt,(int)dPM,sExt[opt1][opt2]);
          sprintf(sTmp,"%s%s",sPathData,sFileOut);

          fprintf(stdout,"%s",sTmp);

          if ( (pfResXX = fopen(sTmp,"rt")) == NULL)
          {
              fprintf(stdout," Resume is not ready!\n");
              continue;
          }

          fprintf(pfRes,"%s\n",sTmp);
          fgets(sLine,BUFSIZ,pfResXX);
          while(!feof(pfResXX))
          {
            if (fgets(sLine,BUFSIZ,pfResXX))
              fprintf(pfRes,"%s",sLine);
          }
//            fprintf(pfRes,"\n");

          fprintf(stdout,"\n",sTmp);

          fclose(pfResXX);
        }
      }

      fclose(pfRes);
    }
  }

  return 0; /* ok */
}
```

Programa 17: Módulo *resume* implementado pelo método *ModuleResume*() da classe *CSystem*

O conteúdo desse arquivo resumo é proveniente dos arquivos resumos de algoritmos gerados durante a execução do módulo *batch*. Esses arquivos resumos tem o seguinte :

$$dsetd\%1dM\%03d\%s.res$$

onde os caracteres de controle %1d, %03d e %s, são substituídos seqüencialmente por d, PM e ALG. Esse último obedece o padrão de extensões de arquivos da Tabela 2.

Então, o arquivo resumo, gerado pelo módulo *resume* nada mais é que uma cópia dos arquivos de resumos menores gerados anteriormente pelo módulo *batch*.

8 Conclusão

Neste trabalho é apresentado algumas abordagens para resolver o problema da mochila, bem como uma prova de que o problema pertence a classe *NP*-completo. Dessa forma, sabe-se que não existe algorito determinista capaz de gerar uma solução ótima em tempo polinomial para o problema, ou seja, um algoritmo determinista em tempo polinomial para resolver o problema. Entretanto, dada as particularidades das instâncias usadas, os algoritmos de Programação Dinâmica, expostos, conseguiram resolver os problemas, dar soluções ótimas, em menos de 1 segundo de processamento. Já os algoritmos expostos relativos a abordagem *Backtracking* não conseguiram varrer todo o espaço de estados, para um conjunto restrito de instâncias, todavia, obtendo soluções que eram compatíveis com as ótimas (mesmas funções objetivos). As abordagens *Greedy* apresentadas mostraram erro relativo inferior a 5%, para todas as instâncias, sendo que a abordagem que usa os objetos ordenados segundo a razão entre a utilidade e o peso apresentaram erro relativo inferior a 1%.

É apresentado, também, um sistema para automatizar todas as tarefas realizadas no trabalho. Uma Grande variedade de tabelas referentes aos experimentos realizados foi anexada ao trabalho, bem como todo o código fonte em *C* utilizado.

9 Anexos

9.1 Tabelas dos Experimentos

Nessa subção estão anexadas as tabelas que resultaram do experimentos sobre as 2000 instâncias. Cada algoritmo implementado para as abordagens, geravam 2000 soluções. Sendo um total de 7 algoritmos implementados para o problema, essas tabelas condensam dados de 14000 experimentos.

Tabela 4: Tabelas com os resultados dos experimentos para o conjunto de dados 1 e tamanho percentual da mochila, $PM = 10\%$.

../data/dset1M010.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M010.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M010.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	138.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	645.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.74Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	4.44Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	7.66Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	53.24Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	14	362.05Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	46	1.09Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	104	2.42Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	209	4.85Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M010.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	125.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	422.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	999.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	1.82Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	3.06Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	16.43Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	5	126.85Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	18	383.58Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	38	817.80Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	74	1.59Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M010.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	10	0	30.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	0	0	0.00 b	4	0	60.00 b	0	0	0.00 b	6	0	60.00 b	1.74%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	2.66%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	2.28%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	3.24%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	3.25%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	3.92%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	4.35%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	4.28%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	4.45%

../data/dset1M010.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	2.44%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.59%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.50%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.62%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.58%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.62%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.34%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.42%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.27%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.50Kb	1.37%

../data/dset1M010.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	7	0	30.00 b	0	0	0.00 b	3	0	30.00 b	0.81%
20	0	0	0.00 b	5	0	60.00 b	0	0	0.00 b	5	0	60.00 b	0.23%
30	0	0	0.00 b	6	0	90.00 b	0	0	0.00 b	4	0	90.00 b	0.12%
40	0	0	0.00 b	4	0	120.00 b	0	0	0.00 b	6	0	120.00 b	0.18%
50	0	0	0.00 b	4	0	150.00 b	0	0	0.00 b	6	0	150.00 b	0.06%
100	0	0	0.00 b	3	0	300.00 b	0	0	0.00 b	7	0	300.00 b	0.03%
200	0	0	0.00 b	1	0	600.00 b	0	0	0.00 b	9	0	600.00 b	0.01%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.01%
400	0	0	0.00 b	2	0	1.20Kb	0	0	0.00 b	8	0	1.20Kb	0.01%
500	0	0	0.00 b	1	0	1.50Kb	0	0	0.00 b	9	0	1.50Kb	0.00%

Tabela 5: Tabelas com os resultados dos experimentos para o conjunto de dados 1 e tamanho percentual da mochila, $PM = 20\%$.

../data/dset1M020.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M020.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	1	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M020.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	177.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	982.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	2.97Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	7.12Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	13.20Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	89.64Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	28	637.04Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	82	1.88Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	188	4.31Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	354	8.11Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M020.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	158.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	553.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.39Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	2.63Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	4.57Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	26.94Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	9	198.30Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	29	612.65Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	65	1.38Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	115	2.49Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M020.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	0.94%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.38%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	1.79%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	2.03%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.74%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	2.06%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	2.66%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	2.76%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	2.68%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	2.73%

../data/dset1M020.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	1.78%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.05%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	90.00 b	1.76%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.27%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.65%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.61%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.48%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.43%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.33%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.26%

../data/dset1M020.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	8	0	30.00 b	0	0	0.00 b	2	0	30.00 b	0.10%
20	0	0	0.00 b	6	0	60.00 b	0	0	0.00 b	4	0	60.00 b	0.18%
30	0	0	0.00 b	2	0	90.00 b	0	0	0.00 b	8	0	90.00 b	0.14%
40	0	0	0.00 b	2	0	120.00 b	0	0	0.00 b	8	0	120.00 b	0.08%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.09%
100	0	0	0.00 b	5	0	300.00 b	0	0	0.00 b	5	0	300.00 b	0.02%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.01%
300	0	0	0.00 b	2	0	900.00 b	0	0	0.00 b	8	0	900.00 b	0.01%
400	0	0	0.00 b	3	0	1.20Kb	0	0	0.00 b	7	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

Tabela 6: Tabelas com os resultados dos experimentos para o conjunto de dados 1 e tamanho percentual da mochila, $PM = 30\%$.

../data/dset1M030.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M030.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M030.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	210.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.26Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	3.93Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	9.30Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	17.55Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	120.92Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	38	848.40Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	109	2.48Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	249	5.68Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	478	10.87Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M030.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	171.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	638.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.65Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	3.10Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	5.85Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	32.02Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	10	231.91Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	35	756.60Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	78	1.63Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	141	2.99Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset1M030.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	6	0	30.00 b	0	0	0.00 b	4	0	30.00 b	0.42%
20	0	0	0.00 b	1	0	60.00 b	0	0	0.00 b	9	0	60.00 b	1.04%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	1.01%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.96%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.17%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	300.00 b	1.31%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	600.00 b	1.68%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.81%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.72%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.69%

../data/dset1M030.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	1.74%
20	0	0	0.00 b	1	0	60.00 b	0	0	0.00 b	9	0	60.00 b	1.08%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.45%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.44%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.51%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	300.00 b	1.60%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.40%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.41%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.37%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.25%

../data/dset1M030.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	6	0	30.00 b	0	0	0.00 b	4	0	30.00 b	0.15%
20	0	0	0.00 b	6	0	60.00 b	0	0	0.00 b	4	0	60.00 b	0.04%
30	0	0	0.00 b	4	0	90.00 b	0	0	0.00 b	6	0	90.00 b	0.07%
40	0	0	0.00 b	2	0	120.00 b	0	0	0.00 b	8	0	120.00 b	0.06%
50	0	0	0.00 b	2	0	150.00 b	0	0	0.00 b	8	0	150.00 b	0.04%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.02%
200	0	0	0.00 b	2	0	600.00 b	0	0	0.00 b	8	0	600.00 b	0.01%
300	0	0	0.00 b	2	0	900.00 b	0	0	0.00 b	8	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	1	0	1.50Kb	0	0	0.00 b	9	0	1.50Kb	0.00%

Tabela 7: Tabelas com os resultados dos experimentos para o conjunto de dados 1 e tamanho percentual da mochila, $PM = 50\%$.

```
../data/dset1M050.bkt1.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10   10    0  20.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 20   10    0  40.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 30   10    0  60.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 40   10    0  80.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 50   10    0 100.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
100   10    0 200.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
200   10    1 400.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
300   10    1 600.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
400   10    0 800.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
500   10    0   1.00Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
../data/dset1M050.bkt2.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10   10    0  20.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 20   10    0  40.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 30   10    0  60.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 40   10    0  80.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 50   10    0 100.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
100   10    0 200.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
200   10    0 400.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
300   10    2 600.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
400   10    0 800.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
500   10    1   1.00Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
../data/dset1M050.dyn1.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10   10    0 265.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 20   10    0   1.62Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 30   10    0   5.20Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 40   10    0  12.29Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 50   10    2  23.36Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
100   10    6 161.77Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
200   10   50   1.14Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
300   10  149   3.42Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
400   10  334   7.72Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
500   10  652  14.89Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
../data/dset1M050.dyn2.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10   10    0 184.00 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 20   10    0 664.60 b    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 30   10    0   1.67Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 40   10    0   3.36Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
 50   10    0   5.45Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
100   10    0  28.45Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
200   10   10 204.08Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
300   10   33 711.00Kb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
400   10   72   1.58Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
500   10  138   2.86Mb    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   0.00%
../data/dset1M050.grd1.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10    0    0   0.00 b    8    0  30.00 b    0    0   0.00 b    2    0  30.00 b   0.06%
 20    0    0   0.00 b    2    0  60.00 b    0    0   0.00 b    8    0  60.00 b   0.32%
 30    0    0   0.00 b    1    0  90.00 b    0    0   0.00 b    9    0  90.00 b   0.45%
 40    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 120.00 b   0.49%
 50    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 150.00 b   0.58%
100    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 300.00 b   0.44%
200    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 600.00 b   0.63%
300    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 900.00 b   0.68%
400    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0   1.20Kb   0.68%
500    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0   1.50Kb   0.70%
../data/dset1M050.grd2.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10    0    0   0.00 b    2    0  30.00 b    0    0   0.00 b    8    0  30.00 b   1.36%
 20    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    1  60.00 b   1.33%
 30    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0  90.00 b   1.54%
 40    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 120.00 b   1.43%
 50    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 150.00 b   1.23%
100    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 300.00 b   1.47%
200    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 600.00 b   1.35%
300    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 900.00 b   1.29%
400    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0   1.20Kb   1.31%
500    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0   1.50Kb   1.25%
../data/dset1M050.grd3.res
     |     OTM       |  |      OTWO      |  |      OTO       |  |          SOW          |
 n   num Time  Space     num Time  Space     num Time  Space     num Time  Space     Error
 10    0    0   0.00 b    6    0  30.00 b    0    0   0.00 b    4    0  30.00 b   0.11%
 20    0    0   0.00 b    6    0  60.00 b    0    0   0.00 b    4    0  60.00 b   0.03%
 30    0    0   0.00 b    7    0  90.00 b    0    0   0.00 b    3    0  90.00 b   0.01%
 40    0    0   0.00 b    2    0 120.00 b    0    0   0.00 b    8    0 120.00 b   0.04%
 50    0    0   0.00 b    4    0 150.00 b    0    0   0.00 b    6    0 150.00 b   0.02%
100    0    0   0.00 b    3    0 300.00 b    0    0   0.00 b    7    0 300.00 b   0.01%
200    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0 600.00 b   0.01%
300    0    0   0.00 b    1    0 900.00 b    0    0   0.00 b    9    0 900.00 b   0.00%
400    0    0   0.00 b    2    0   1.20Kb    0    0   0.00 b    8    0   1.20Kb   0.00%
500    0    0   0.00 b    0    0   0.00 b    0    0   0.00 b   10    0   1.50Kb   0.00%
```

Tabela 8: Tabelas com os resultados dos experimentos para o conjunto de dados 2 e tamanho percentual da mochila, $PM = 10\%$.

../data/dset2M010.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M010.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	1	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M010.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	122.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	537.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.74Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	3.65Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	6.32Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	39.85Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	11	250.16Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	31	721.36Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	62	1.45Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	112	2.62Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M010.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	124.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	396.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.03Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	1.80Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	2.96Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	17.52Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	6	106.65Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	13	295.92Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	30	642.11Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	53	1.15Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M010.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	8	0	30.00 b	0	0	0.00 b	2	0	30.00 b	0.60%
20	0	0	0.00 b	3	0	60.00 b	0	0	0.00 b	7	0	60.00 b	1.65%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	2.13%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	3.01%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	3.60%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	4.00%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	3.82%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	4.16%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	4.41%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	4.48%

../data/dset2M010.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	6	0	30.00 b	0	0	0.00 b	4	0	30.00 b	0.71%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	2.04%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.67%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.96%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.27%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.34%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.49%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.59%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.43%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.36%

../data/dset2M010.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	7	0	30.00 b	0	0	0.00 b	3	0	30.00 b	0.41%
20	0	0	0.00 b	4	0	60.00 b	0	0	0.00 b	6	0	60.00 b	0.19%
30	0	0	0.00 b	5	0	90.00 b	0	0	0.00 b	5	0	90.00 b	0.19%
40	0	0	0.00 b	3	0	120.00 b	0	0	0.00 b	7	0	120.00 b	0.14%
50	0	0	0.00 b	2	0	150.00 b	0	0	0.00 b	8	0	150.00 b	0.10%
100	0	0	0.00 b	1	0	300.00 b	0	0	0.00 b	9	0	300.00 b	0.07%
200	0	0	0.00 b	2	0	600.00 b	0	0	0.00 b	8	0	600.00 b	0.01%
300	0	0	0.00 b	2	0	900.00 b	0	0	0.00 b	8	0	900.00 b	0.01%
400	0	0	0.00 b	1	0	1.20Kb	0	0	0.00 b	9	0	1.20Kb	0.01%
500	0	0	0.00 b	2	0	1.50Kb	0	0	0.00 b	8	0	1.50Kb	0.00%

Tabela 9: Tabelas com os resultados dos experimentos para o conjunto de dados 2 e tamanho percentual da mochila, $PM = 20\%$.

../data/dset2M020.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M020.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	1	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	1	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	1	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M020.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	166.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	838.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	2.70Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	6.05Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	10.84Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	2	68.41Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	18	437.16Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	54	1.26Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	108	2.51Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	187	4.34Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M020.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	147.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	509.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.32Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	2.73Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	4.35Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	25.22Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	9	171.07Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	22	476.95Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	48	1.02Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	84	1.79Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M020.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	8	0	30.00 b	0	0	0.00 b	2	0	30.00 b	0.19%
20	0	0	0.00 b	1	0	60.00 b	0	0	0.00 b	9	0	60.00 b	1.15%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.48%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	2.30%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.76%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	2.80%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	600.00 b	2.64%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	2.60%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	2.75%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	2.84%

../data/dset2M020.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	2.45%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.67%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.57%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.91%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.45%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.33%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.35%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.48%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.43%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.32%

../data/dset2M020.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	5	0	30.00 b	0	0	0.00 b	5	0	30.00 b	0.55%
20	0	0	0.00 b	3	0	60.00 b	0	0	0.00 b	7	0	60.00 b	0.29%
30	0	0	0.00 b	4	0	90.00 b	0	0	0.00 b	6	0	90.00 b	0.12%
40	0	0	0.00 b	6	0	120.00 b	0	0	0.00 b	4	0	120.00 b	0.04%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.04%
100	0	0	0.00 b	1	0	300.00 b	0	0	0.00 b	9	0	300.00 b	0.03%
200	0	0	0.00 b	1	0	600.00 b	0	0	0.00 b	9	0	600.00 b	0.01%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.01%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.01%
500	0	0	0.00 b	1	0	1.50Kb	0	0	0.00 b	9	0	1.50Kb	0.00%

Tabela 10: Tabelas com os resultados dos experimentos para o conjunto de dados 2 e tamanho percentual da mochila, $PM = 30\%$.

../data/dset2M030.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	1	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M030.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M030.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	201.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.07Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	3.48Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	8.32Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	14.81Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	4	89.81Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	26	586.54Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	70	1.67Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	139	3.28Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	245	5.66Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M030.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	170.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	574.60 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.57Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	3.48Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	5.43Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	29.75Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	9	208.08Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	25	560.40Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	54	1.18Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	98	2.15Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M030.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	5	0	30.00 b	0	0	0.00 b	5	0	30.00 b	0.39%
20	0	0	0.00 b	3	0	60.00 b	0	0	0.00 b	7	0	60.00 b	0.95%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	1.01%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.45%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.13%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.42%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.49%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.69%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.66%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.71%

../data/dset2M030.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.82%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.43%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.52%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.59%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.33%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.38%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.27%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.45%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.43%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.39%

../data/dset2M030.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	0.29%
20	0	0	0.00 b	5	0	60.00 b	0	0	0.00 b	5	0	60.00 b	0.12%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	0.11%
40	0	0	0.00 b	3	0	120.00 b	0	0	0.00 b	7	0	120.00 b	0.06%
50	0	0	0.00 b	3	0	150.00 b	0	0	0.00 b	7	0	150.00 b	0.04%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.03%
200	0	0	0.00 b	2	0	600.00 b	0	0	0.00 b	8	1	600.00 b	0.01%
300	0	0	0.00 b	2	0	900.00 b	0	0	0.00 b	8	0	900.00 b	0.01%
400	0	0	0.00 b	1	0	1.20Kb	0	0	0.00 b	9	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

Tabela 11: Tabelas com os resultados dos experimentos para o conjunto de dados 2 e tamanho percentual da mochila, $PM = 50\%$.

../data/dset2M050.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	1	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	2	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M050.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	1	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	1	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M050.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	251.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	1	1.42Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	4.63Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	11.34Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	2	20.30Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	123.87Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	33	782.07Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	92	2.20Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	181	4.33Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	312	7.37Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M050.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	190.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	627.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	1.55Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	3.43Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	5.25Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	28.64Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	9	180.07Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	25	515.78Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	50	1.09Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	90	1.95Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset2M050.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	0.69%
20	0	0	0.00 b	5	0	60.00 b	0	0	0.00 b	5	0	60.00 b	0.33%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	0.35%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.53%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.47%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.51%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.56%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.68%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.74%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.72%

../data/dset2M050.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	1.38%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.35%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.33%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.69%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.31%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.27%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.26%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.30%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.30%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.50Kb	1.30%

../data/dset2M050.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	8	0	30.00 b	0	0	0.00 b	2	0	30.00 b	0.04%
20	0	0	0.00 b	3	0	60.00 b	0	0	0.00 b	7	0	60.00 b	0.14%
30	0	0	0.00 b	5	0	90.00 b	0	0	0.00 b	5	0	90.00 b	0.03%
40	0	0	0.00 b	5	0	120.00 b	0	0	0.00 b	5	0	120.00 b	0.04%
50	0	0	0.00 b	3	0	150.00 b	0	0	0.00 b	7	0	150.00 b	0.02%
100	0	0	0.00 b	2	0	300.00 b	0	0	0.00 b	8	0	300.00 b	0.01%
200	0	0	0.00 b	1	0	600.00 b	0	0	0.00 b	9	0	600.00 b	0.01%
300	0	0	0.00 b	1	0	900.00 b	0	0	0.00 b	9	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	1	0	1.50Kb	0	0	0.00 b	9	0	1.50Kb	0.00%

Tabela 12: Tabelas com os resultados dos experimentos para o conjunto de dados 3 e tamanho percentual da mochila, $PM = 10\%$.

../data/dset3M010.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	6	807	400.00 b	0	0	0.00 b	4	4990	400.00 b	0	0	0.00 b	0.00%
300	6	1300	600.00 b	0	0	0.00 b	4	4995	600.00 b	0	0	0.00 b	0.00%
400	5	934	800.00 b	0	0	0.00 b	5	4992	800.00 b	0	0	0.00 b	0.00%
500	8	132	1.00Kb	0	0	0.00 b	2	4990	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M010.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	7	1003	400.00 b	0	0	0.00 b	3	4993	400.00 b	0	0	0.00 b	0.00%
300	8	1434	600.00 b	0	0	0.00 b	2	4990	600.00 b	0	0	0.00 b	0.00%
400	6	878	800.00 b	0	0	0.00 b	4	4995	800.00 b	0	0	0.00 b	0.00%
500	9	511	1.00Kb	0	0	0.00 b	1	5000	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M010.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	168.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.41Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	4.31Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	9.54Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	16.48Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	75.59Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	10	316.80Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	24	717.87Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	45	1.29Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	72	2.02Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M010.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	110.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	634.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	2.43Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	5.92Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	11.09Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	62.50Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	11	287.96Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	26	684.10Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	48	1.25Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	73	2.00Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M010.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	1.23%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	2.48%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	2.30%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	2.61%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	2.85%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	3.02%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	3.13%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	3.14%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	3.12%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	3.14%

../data/dset3M010.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	1.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.11%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	0.78%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.62%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.39%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.14%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.12%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.07%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.05%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

../data/dset3M010.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	1.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.11%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	0.78%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.62%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.39%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.14%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.12%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.07%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.05%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

Tabela 13: Tabelas com os resultados dos experimentos para o conjunto de dados 3 e tamanho percentual da mochila, $PM = 20\%$.

../data/dset3M020.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	11	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	8	1342	400.00 b	0	0	0.00 b	2	5000	400.00 b	0	0	0.00 b	0.00%
300	8	1234	600.00 b	0	0	0.00 b	2	4995	600.00 b	0	0	0.00 b	0.00%
400	3	1010	800.00 b	0	0	0.00 b	7	4997	800.00 b	0	0	0.00 b	0.00%
500	4	1868	1.00Kb	0	0	0.00 b	6	5000	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M020.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	8	708	400.00 b	0	0	0.00 b	2	5000	400.00 b	0	0	0.00 b	0.00%
300	8	650	600.00 b	0	0	0.00 b	2	4990	600.00 b	0	0	0.00 b	0.00%
400	3	530	800.00 b	0	0	0.00 b	7	4991	800.00 b	0	0	0.00 b	0.00%
500	4	1015	1.00Kb	0	0	0.00 b	6	4992	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M020.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	297.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	2.88Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	9.12Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	18.05Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	30.36Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	135.65Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	19	564.34Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	46	1.28Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	82	2.30Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	128	3.59Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M020.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	179.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.71Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	1	6.46Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	13.73Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	24.33Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	121.71Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	20	536.48Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	47	1.25Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	88	2.28Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	139	3.61Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M020.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.84%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.99%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.68%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	2.09%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	2.18%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	2.24%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	2.30%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	2.30%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	2.29%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	2.29%

../data/dset3M020.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	0.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.97%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.67%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.49%
50	0	0	0.00 b	2	0	150.00 b	0	0	0.00 b	8	0	150.00 b	0.30%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.14%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.07%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.04%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.04%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

../data/dset3M020.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	4	0	30.00 b	0	0	0.00 b	6	0	30.00 b	0.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.97%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.67%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.49%
50	0	0	0.00 b	2	0	150.00 b	0	0	0.00 b	8	0	150.00 b	0.30%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.14%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.07%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	900.00 b	0.04%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.04%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

Tabela 14: Tabelas com os resultados dos experimentos para o conjunto de dados 3 e tamanho percentual da mochila, $PM = 30\%$.

../data/dset3M030.bkt1.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	2	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	252	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	7	667	400.00 b	0	0	0.00 b	3	4997	400.00 b	0	0	0.00 b	0.00%
300	4	5	600.00 b	0	0	0.00 b	6	4995	600.00 b	0	0	0.00 b	0.00%
400	1	0	800.00 b	0	0	0.00 b	9	4992	800.00 b	0	0	0.00 b	0.00%
500	3	3	1.00Kb	0	0	0.00 b	7	4996	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M030.bkt2.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	134	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	7	351	400.00 b	0	0	0.00 b	3	4993	400.00 b	0	0	0.00 b	0.00%
300	4	2	600.00 b	0	0	0.00 b	6	4990	600.00 b	0	0	0.00 b	0.00%
400	1	0	800.00 b	0	0	0.00 b	9	4994	800.00 b	0	0	0.00 b	0.00%
500	4	1058	1.00Kb	0	0	0.00 b	6	4997	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M030.dyn1.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	437.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	4.16Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	13.24Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	25.19Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	2	41.39Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	7	184.82Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	29	765.59Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	61	1.73Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	111	3.12Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	174	4.88Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M030.dyn2.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	262.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	2.89Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	9.84Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	19.99Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	34.92Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	8	167.05Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	27	724.08Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	66	1.68Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	121	3.05Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	186	4.81Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M030.grd1.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.31%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.54%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.43%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.69%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.79%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.80%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.84%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.83%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.82%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.82%

../data/dset3M030.grd2.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	0.88%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.61%
30	0	0	0.00 b	2	0	90.00 b	0	0	0.00 b	8	0	90.00 b	0.30%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.33%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.33%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.15%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.06%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.05%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.05%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

../data/dset3M030.grd3.res

n	OTM			OTWO			OTO			SOW			
	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	0.88%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.61%
30	0	0	0.00 b	2	0	90.00 b	0	0	0.00 b	8	0	90.00 b	0.30%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.33%
50	0	0	0.00 b	1	0	150.00 b	0	0	0.00 b	9	0	150.00 b	0.33%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.15%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.06%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.05%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.05%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.03%

Tabela 15: Tabelas com os resultados dos experimentos para o conjunto de dados 3 e tamanho percentual da mochila, $PM = 50\%$.

../data/dset3M050.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	4	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	3	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	9	368	200.00 b	0	0	0.00 b	1	5000	200.00 b	0	0	0.00 b	0.00%
200	4	0	400.00 b	0	0	0.00 b	6	4998	400.00 b	0	0	0.00 b	0.00%
300	6	513	600.00 b	0	0	0.00 b	4	4998	600.00 b	0	0	0.00 b	0.00%
400	8	182	800.00 b	0	0	0.00 b	2	4995	800.00 b	0	0	0.00 b	0.00%
500	4	440	1.00Kb	0	0	0.00 b	6	4993	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M050.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	1	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	4	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	9	206	200.00 b	0	0	0.00 b	1	5000	200.00 b	0	0	0.00 b	0.00%
200	4	0	400.00 b	0	0	0.00 b	6	5000	400.00 b	0	0	0.00 b	0.00%
300	7	844	600.00 b	0	0	0.00 b	3	4990	600.00 b	0	0	0.00 b	0.00%
400	8	92	800.00 b	0	0	0.00 b	2	4990	800.00 b	0	0	0.00 b	0.00%
500	4	245	1.00Kb	0	0	0.00 b	6	4997	1.00Kb	0	0	0.00 b	0.00%

../data/dset3M050.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	641.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	6.06Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	1	19.34Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	35.45Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	57.83Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	8	257.10Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	40	1.06Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	85	2.40Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	150	4.32Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	238	6.77Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M050.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	391.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	1	4.21Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	1	13.64Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	27.01Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	3	46.32Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	9	216.10Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	36	926.97Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	86	2.13Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	155	3.86Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	239	6.09Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset3M050.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.08%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.12%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.99%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	1.17%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	1.23%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	1.23%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	1.27%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	1.27%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	1.26%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	1.25%

../data/dset3M050.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	2	0	30.00 b	0	0	0.00 b	8	0	30.00 b	0.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.58%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.25%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.33%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.24%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.11%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.05%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.02%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.01%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.02%

../data/dset3M050.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	2	0	30.00 b	0	0	0.00 b	8	0	30.00 b	0.69%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.58%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.25%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.33%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.24%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.11%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.05%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	900.00 b	0.02%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.01%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.02%

Tabela 16: Tabelas com os resultados dos experimentos para o conjunto de dados 4 e tamanho percentual da mochila, $PM = 10\%$.

../data/dset4M010.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	2	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	1	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M010.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	1	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M010.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	154.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.40Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	5.34Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	11.11Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	18.90Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	3	87.01Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	12	356.87Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	30	791.17Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	49	1.43Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	80	2.22Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M010.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	113.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	609.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	2.44Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	5.75Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	10.44Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	60.89Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	11	289.43Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	25	673.75Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	48	1.25Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	77	1.96Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M010.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	0.73%
20	0	0	0.00 b	1	0	60.00 b	0	0	0.00 b	9	0	60.00 b	0.41%
30	0	0	0.00 b	1	0	90.00 b	0	0	0.00 b	9	0	90.00 b	0.13%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.18%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.12%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.10%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.11%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.11%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.10%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.10%

../data/dset4M010.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	1.71%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.60%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	1.12%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.70%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.75%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.43%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.22%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.19%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.20Kb	0.18%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.16%

../data/dset4M010.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	2	0	30.00 b	0	0	0.00 b	8	0	30.00 b	1.24%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.63%
30	0	0	0.00 b	2	0	90.00 b	0	0	0.00 b	8	0	90.00 b	0.29%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.22%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.11%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	300.00 b	0.03%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.01%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.01%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.50Kb	0.00%

Tabela 17: Tabelas com os resultados dos experimentos para o conjunto de dados 4 e tamanho percentual da mochila, $PM = 20\%$.

../data/dset4M020.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M020.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M020.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	310.20 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	3.74Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	11.96Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	22.59Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	2	37.12Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	7	164.34Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	23	666.26Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	54	1.48Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	96	2.67Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	148	4.16Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M020.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	187.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	1.63Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	6.28Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	13.06Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	22.82Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	5	116.92Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	20	528.23Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	47	1.21Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	89	2.21Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	139	3.47Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M020.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	0.40%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.18%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.09%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.09%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.08%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.08%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.09%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.09%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.08%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.08%

../data/dset4M020.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	2.28%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	1.14%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.75%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.54%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.51%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.27%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.20%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.14%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.20Kb	0.13%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.12%

../data/dset4M020.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.81%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.30%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.15%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.17%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.09%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.01%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.01%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

Tabela 18: Tabelas com os resultados dos experimentos para o conjunto de dados 4 e tamanho percentual da mochila, $PM = 30\%$.

../data/dset4M030.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	1	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M030.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	1	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M030.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	502.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	6.05Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	1	17.99Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	2	32.86Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	53.42Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	8	235.13Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	34	942.94Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	76	2.10Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	133	3.79Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	209	5.89Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M030.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	264.40 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	1	2.63Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	2	9.39Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	18.61Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	31.88Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	7	156.60Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	29	694.54Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	65	1.58Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	118	2.88Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	183	4.51Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M030.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	2	0	30.00 b	0	0	0.00 b	8	0	30.00 b	0.25%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.10%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.08%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.08%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.06%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.06%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.07%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.07%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	1.20Kb	0.06%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.06%

../data/dset4M030.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.33%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.95%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.51%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.66%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.31%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.25%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.14%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.11%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.11%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.11%

../data/dset4M030.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	1	0	30.00 b	0	0	0.00 b	9	0	30.00 b	0.73%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.26%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.16%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.04%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.05%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.01%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.00%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

Tabela 19: Tabelas com os resultados dos experimentos para o conjunto de dados 4 e tamanho percentual da mochila, $PM = 50\%$.

../data/dset4M050.bkt1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	0	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	1	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M050.bkt2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	20.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	40.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	60.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	0	80.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	100.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	0	200.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	1	400.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	0	600.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	0	800.00 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	0	1.00Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M050.dyn1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	878.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	9.96Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	1	28.01Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	2	50.12Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	0	80.93Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	12	353.49Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	50	1.41Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	112	3.14Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	199	5.66Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	315	8.81Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M050.dyn2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	10	0	363.80 b	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
20	10	0	3.60Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
30	10	0	12.12Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
40	10	1	23.41Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
50	10	1	39.41Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
100	10	6	189.02Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
200	10	34	830.35Kb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
300	10	76	1.88Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
400	10	139	3.42Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%
500	10	217	5.36Mb	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	0.00%

../data/dset4M050.grd1.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	2	0	30.00 b	0	0	0.00 b	8	1	30.00 b	0.23%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.06%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.05%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.05%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.05%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	1	300.00 b	0.04%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.05%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.05%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.05%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.05%

../data/dset4M050.grd2.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	30.00 b	1.41%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.63%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.54%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.41%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.31%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.13%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.12%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.11%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.07%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.08%

../data/dset4M050.grd3.res

	OTM			OTWO			OTO			SOW			
n	num	Time	Space	num	Time	Space	num	Time	Space	num	Time	Space	Error
10	0	0	0.00 b	3	0	30.00 b	0	0	0.00 b	7	0	30.00 b	0.35%
20	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	60.00 b	0.10%
30	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	90.00 b	0.07%
40	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	120.00 b	0.05%
50	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	150.00 b	0.04%
100	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	300.00 b	0.01%
200	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	600.00 b	0.00%
300	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	900.00 b	0.00%
400	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.20Kb	0.00%
500	0	0	0.00 b	0	0	0.00 b	0	0	0.00 b	10	0	1.50Kb	0.00%

9.2 Códigos fontes

Os códigos desta seção estão anexados em ordem de relevância e de entendimento do trabalho.

```
# Sistema Unix, compilador gnu
CC = /usr/bin/g++

LIBS = -lm

# Diretorio raiz
T = .

#Diretorio dos arquivos .h
INCLUDE = $(T)/

# Opcoes de compilacao
#OPTIONST = -c -g -ansi -O3 -Wall -I $(INCLUDE)
OPTIONST = -c -O3 -Wall -I $(INCLUDE)

# Insira aqui o diretorio dos arquivos objetos sem a barra no final
obj_T = $(T)

# Nome do arquivo executavel
EXE=$(T)/knapsack

 all: f1
   $(CC) -o $(EXE)      $(obj_T)/*.o $(LIBS)
   rm -rf $(obj_T)/*.o

 f1:
   $(CC) $(OPTIONST) $(T)/util.cpp          -o $(obj_T)/util.o
   $(CC) $(OPTIONST) $(T)/heapsort.cpp      -o $(obj_T)/heapsort.o
   $(CC) $(OPTIONST) $(T)/knapsack.cpp      -o $(obj_T)/knapsack.o
   $(CC) $(OPTIONST) $(T)/system.cpp        -o $(obj_T)/system.o
   $(CC) $(OPTIONST) $(T)/main.cpp -o $(obj_T)/main.o
run:
   $(EXE)


runb:  all
   $(EXE) file 1 1 ./avaliacao.tp2 >saidab
rund:  all
   $(EXE) file 2 1 ./avaliacao.tp2 >saidad
compg: all

 clean:
   rm -rf $(obj_T)/*.o
   rm -rf $(EXE)
```

Programa 18: Arquivo makefile

```
#ifndef _TDA_H_
#define _TDA_H_

/* inserir o código do
Programa 1 aqui */
```

```
#endif
```

Programa 19: Arquivo tda.h

```
#ifndef _KNAPSACK_H_
#define _KNAPSACK_H_

#include "tda.h"
#include "stacktime.h"

void Estouro(int sig);

/* inserir o código do Programa 2 aqui */
#endif
```

Programa 20: Arquivo knapsack.h

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
#include <unistd.h>

#include "util.h"
#include "heapsort.h"

#include "knapsack.h"

int g_bOverflow; /* global somente neste modulo */

void Overflow(int sig)
{
  g_bOverflow=TRUE;
  alarm(0);
  fprintf(stderr,"Overflow! ");
}

CKnapSack::CKnapSack()
{
  m_iM = 0;
  m_in = 0;

  m_iOverflowTime = 5;

  InitData();
  InitDataSolution();
  InitDataBacktracking();
  InitDataDynamicProgramming();

  /* signal that must be sended in overflow case */
  signal(SIGALRM,Overflow);
}

CKnapSack::CKnapSack(int _M,int _n,Vector _Data)
{
  m_iM = _M;
  m_in = _n;
```

```
  InitData();
  InitDataSolution();
  InitDataBacktracking();
  InitDataDynamicProgramming();

  CreateData();

  for(int i = 1 ; i <= m_in ; i++)
    m_pData[i] = _Data[i];

  /* signal that must be sended in overflow case */
  signal(SIGALRM, Overflow);
}

CKnapSack::~CKnapSack()
{
  ClearData();
  ClearDataSolution();
  ClearDataBacktracking();
  ClearDataDynamicProgramming();
}

void CKnapSack::InitData()
{
  m_pData = NULL;
}

void CKnapSack::InitDataSolution()
{
  m_piX     = NULL;
}

void CKnapSack::InitDataBacktracking()
{
  m_piY = NULL;
}

void CKnapSack::InitDataDynamicProgramming()
{
  m_piPLeft = NULL;
  m_piPL    = NULL;
  m_ppData  = NULL;
  m_piSize  = NULL;
  m_piSizeM = NULL;
}

void CKnapSack::CreateData()
{
  ClearData();

  m_pData = new Object[m_in+1];
}

void CKnapSack::CreateDataSolution()
{
  ClearDataSolution();

  m_piX       = new int[m_in+1];
```

```
  for(int i=1;i<=m_in; i++)
    m_piX[i] = 0;
}

void CKnapSack::StartProcess()
{
  StartTime();

  /* configura alarme */
  alarm(m_iOverflowTime);

  m_ifp = 0;
  m_ifw = 0;

  m_bOptimal   =FALSE;
  m_bSolution =FALSE;
  g_bOverflow =FALSE;
  m_bOverflow =FALSE;
}

void CKnapSack::CreateDataBacktracking()
{
  ClearDataBacktracking();

  m_piY = new int[m_in+1];

  for(int i=1;i<=m_in; i++)
    m_piY[i] = 0;
}

void CKnapSack::CreateDataDynamicProgramming()
{
  ClearDataDynamicProgramming();

  m_piPLeft = new int[m_in+1];
  m_piPL     = new int[m_in+1];
  m_ppData   = new Object*[m_in+1];
  m_piSize   = new int[m_in+1];
  m_piSizeM = new int[m_in+1];

  /* for dynamic programming */
  for(int i = 0 ; i <= m_in ; i++)
  {
    m_piPLeft[i] = 0;
    m_piPL[i]      = 0;
    m_ppData[i]    =NULL;
    m_piSize[i]    = 0;
    m_piSizeM[i] = 0;
  }
}

void CKnapSack::ClearData()
{
  if (m_pData != NULL) {delete m_pData; m_pData = NULL;}
}

void CKnapSack::ClearDataSolution()
{
```

```
  if (m_piX    != NULL) {delete m_piX;    m_piX    = NULL;}
}

void CKnapSack::ClearDataBacktracking()
{
  if (m_piY != NULL) {delete []m_piY; m_piY = NULL;}
}

void CKnapSack::ClearDataDynamicProgramming()
{
  if (m_ppData != NULL)
  {
    for(int i = 0 ; i <= m_in ; i++)
        if (m_ppData[i] != NULL){ delete m_ppData[i]; m_ppData[i] = NULL; }
    delete []m_ppData; m_ppData = NULL;
  }

  if (m_piPLeft != NULL) delete []m_piPLeft;  m_piPLeft = NULL;
  if (m_piPL    != NULL) delete []m_piPL;     m_piPL    = NULL;
  if (m_piSize  != NULL) delete []m_piSize;   m_piSize  = NULL;
  if (m_piSizeM != NULL) delete []m_piSizeM;  m_piSizeM = NULL;
}

void CKnapSack::Setn(int _n)
{
  ClearData();
  ClearDataSolution();
  ClearDataBacktracking();
  ClearDataDynamicProgramming();

  m_in = _n;
  m_iM = 0;

  CreateData();
}

void CKnapSack::SetPM(double dPM)
{
  long double sumW=0;
  for( int i = 1 ; i <= m_in ; i++ )
    sumW += m_pData[i].W;

  m_iM = (int)(dPM*sumW/100+0.5); /* 0.5 para arredondamento */
}

void CKnapSack::SetData(Vector _Data)
{
  ClearData();
  CreateData();

  for(int i = 1 ; i <= m_in ; i++)
    m_pData[i] = _Data[i];
}

void CKnapSack::SetDataSolution(int i,int data)
{
  if (m_piX==NULL) CreateDataSolution();
```

```
    if ( (i >= 1) && (i<=m_in) )
        m_piX[i] = data;
}

int CKnapSack::Getn()
{
    return m_in;
}

int CKnapSack::GetM()
{
    return m_iM;
}

int CKnapSack::Getfw()
{
    return m_ifw;
}

int CKnapSack::Getfp()
{
    return m_ifp;
}

int CKnapSack::GetDataP(int i)
{
    return ((i >= 1)&&(i<=m_in)) ? m_pData[i].P : 0;
}

int CKnapSack::GetDataW(int i)
{
    return ((i >= 1)&&(i<=m_in)) ? m_pData[i].W : 0;
}

int CKnapSack::GetDataSolution(int i)
{
    return ((i >= 1)&&(i<=m_in)) ? m_piX[i] : -1;
}

void CKnapSack::StartTime()
{
    m_oStackTime.Start();
    m_iTime   = 0;
}

void CKnapSack::FinishTime()
{
    m_oStackTime.Finish();
    m_iTime   = (int)(1000*m_oStackTime.GetElapTime()+0.5);
}

void CKnapSack::ShowData(FILE *pF)
{
    fprintf(pF,"n = %d, M= %d\n",m_in,m_iM);
    fprintf(pF,"W P\n");
    for( int i = 1 ; i <= m_in ; i++ )
        fprintf(pF,"%d %d\n",m_pData[i].W,m_pData[i].P);
}
```

```
void CKnapSack::ShowDatanM(FILE* pF)
{
  fprintf(pF,"n = %d, M= %d\n",m_in,m_iM);
}
void CKnapSack::ShowDataSolution(FILE *pF)
{
    int iCt=0,i;
    Vector pDataTemp = new Object[m_in+1];

  for( i = 1 ; i <= m_in ; i++ )
    if (m_piX[i])
      pDataTemp[++iCt] = m_pData[i];

  GenerateKeyByRatio();
  Heapsort(pDataTemp,&iCt,0);

  for( i = 1 ; i <= iCt ; i++ )
      fprintf(pF,"%d %d\n",pDataTemp[i].W,pDataTemp[i].P);

  delete[] pDataTemp;
}

void CKnapSack::ShowDataBacktracking(FILE *pF)
{ }

void CKnapSack::ShowDataDynamicProgramming(FILE *pF)
{ }

void CKnapSack::PrintState(FILE* pF)
{
  fprintf(pF,"Optimal: %d, Solution: %d, Overflow: %d, ww: %d, pp: %d\n"
    ,IsOptimal()
    ,IsSolution()
    ,IsOverflow()
    ,Getfw()
    ,Getfp());
}

void CKnapSack::VerifyResult()
{
  int pp=0,ww=0,i;
  for( i = 1 ; i <= m_in ; i++ )
  {
    pp += m_piX[i]*m_pData[i].P;
    ww += m_piX[i]*m_pData[i].W;
  }
  if ( (pp != m_ifp) || (ww != m_ifw) )
  {
    fprintf(stderr
      ,"solution informed not corresponding to profit and weight computed!\n");
    fprintf(stderr
      ,"computer: pp = %d, ww= %d\n",pp,ww);
    ShowData(stdout);
    exit(0);
  }
}
```

```
/* inserir o código do Programa 6 aqui */

/* inserir o código do Programa 8 aqui */

/* inserir o código do Programa 7 aqui */

/* inserir o código do Programa 9 aqui */

void CKnapSack::CreateVectorPW(int n)
{
  if ( (m_ppData[n] == NULL) )
  {
    /* in worst case the number of objects is 2*|S^{i-1}| */
    /* |S^i| <= 2|S^{i-1}| */
    /* + 1 adaptation for C language */
    m_piSizeM[n] = ((n!=0)?2*m_piSize[n-1]:1) + 1;

    m_ppData[n] = new Object[ ((n!=0)?2*m_piSize[n-1]:1) + 1];
    m_ppData[n][0].P = ((n!=0)?m_ppData[n-1][m_piSize[n-1]].P:0);
    m_ppData[n][0].W = ((n!=0)?m_ppData[n-1][m_piSize[n-1]].W:0);

    m_piSize[n] = 0;
  }
}

void CKnapSack::AddPW(int n,int _P,int _W)
{
  CreateVectorPW(n);

  m_piSize[n]++;

  m_ppData[n][m_piSize[n]].P = _P;
  m_ppData[n][m_piSize[n]].W = _W;
}

/* inserir o código do Programa 12 aqui */

/* inserir o código do Programa 10 aqui */

/* inserir o código do Programa 11 aqui */

/* inserir o código do Programa 13 aqui */

/* Generate Key by Ratio for sort process */
void CKnapSack::GenerateKeyByRatio(Vector pData,int n)
{
  for( int i = 1 ; i <= n ; i++ )
    pData[i].Key = (double)pData[i].P/pData[i].W;
}

/* Generate Key by Ratio for sort process */
void CKnapSack::GenerateKeyByRatio()
{
  for( int i = 1 ; i <= m_in ; i++ )
    m_pData[i].Key = (double)m_pData[i].P/m_pData[i].W;
}

/* Generate Key by Profit (P) for sort process */
```

```
void CKnapSack::GenerateKeyByProfit()
{
  for( int i = 1 ; i<=m_in ; i++)
    m_pData[i].Key = (double)m_pData[i].P;
}

/* Generate Key by Weight (W) for sort process */
void CKnapSack::GenerateKeyByWeight()
{
  for( int i = 1 ; i<=m_in ; i++)
    m_pData[i].Key = (double)m_pData[i].W;
}

/* Save Instance in File */
void CKnapSack::FileSaveInstance(char* sFileName)
{
  int i;
  FILE *pfOut;

  if ( (pfOut=fopen(sFileName,"wt+")) ==NULL )
  {
    fprintf(stderr,"error generating instance!\n");
    exit(0);
  }
  fprintf(pfOut,"%d %d\n",m_in,m_iM);

  for( i = 1 ; i<=m_in ; i++)
    fprintf(pfOut,"%4d %4d\n",m_pData[i].W,m_pData[i].P);
  fprintf(pfOut,"\n");
  fprintf(pfOut,"n M\n");
  fprintf(pfOut,"W1 P1\n");
  fprintf(pfOut," .   .\n");
  fprintf(pfOut," .   .\n");
  fprintf(pfOut," .   .\n");
  fprintf(pfOut,"Wn Pn\n");

  fclose(pfOut);
}

/* Load Instance from file */
int CKnapSack::FileLoadInstance(char* sFileName)
{
  int i;
  FILE *pfOut;

  if ( (pfOut=fopen(sFileName,"rt")) ==NULL )
    return 0; /* error, file not found */

  /* clear data */
  ClearData();
  ClearDataSolution();
  ClearDataBacktracking();
  ClearDataDynamicProgramming();

  fscanf(pfOut,"%d %d\n",&m_in,&m_iM);

  CreateData();
```

```
  for( i = 1 ; i<=m_in ; i++)
    fscanf(pfOut,"%d %d\n",&(m_pData[i].W),&(m_pData[i].P) );
  fscanf(pfOut,"\n");

  fclose(pfOut);
  return 1; /* OK - success */
}

/* Save Solution to file */
void CKnapSack::FileSaveSolution(char* sFileName)
{
  int i;
  FILE *pfOut;

  if ( (pfOut=fopen(sFileName,"wt+")) == NULL )
  {
    fprintf(stderr,"error generating solution file!\n");
    exit(0);
  }
  fprintf(pfOut,"%d %d %d %d %d\n",m_in,m_iM,m_bOptimal,m_bSolution,m_bOverflow);
  for( i = 1 ; i<=m_in ; i++)
    if (m_piX[i])
      fprintf(pfOut,"%d %d\n",m_pData[i].W,m_pData[i].P);

  fclose(pfOut);
}

/* Load Solution from file */
int CKnapSack::FileLoadSolution(char* sFileName)
{
    int i;
  Object Data;
  FILE *pfOut;

  /* Clear Old Data */
  ClearDataSolution();
  ClearDataBacktracking();
  ClearDataDynamicProgramming();

  m_ifw = 0;
  m_ifp = 0;

  m_bOptimal   =FALSE;
  m_bSolution = FALSE;
  g_bOverflow = FALSE;
  m_bOverflow = FALSE;

  if ( (pfOut=fopen(sFileName,"rt")) == NULL )
    return 0; /* error, file not found */

  fscanf(pfOut
    ,"%d %d %d %d %d\n",&m_in,&m_iM,&m_bOptimal,&m_bSolution,&m_bOverflow);

  CreateDataSolution();

  int nitems;
  do
  {
```

```
    nitems=fscanf(pfOut,"%d %d\n",&(Data.W),&(Data.P));
    for( i = 1 ; i <=m_in && (nitems == 2); i++ )
      if ( (Data.W==m_pData[i].W) && (Data.P ==m_pData[i].P)  &&
           (m_piX[i] == 0) )
      {
        m_piX[i] = 1;
        break;
      }
  } while(nitems == 2);

  for( i = 1 ; i <=m_in; i++ )
  {
    m_ifw +=m_piX[i] * m_pData[i].W;
    m_ifp +=m_piX[i] * m_pData[i].P;
  }

  fclose(pfOut);
  return 1; /* OK- success */
}

/* inserir o código do Programa 4 aqui */
```

Programa 21: Arquivo knapsack.cpp

```
#ifndef _SYSTEM_H_
#define _SYSTEM_H_

#include "knapsack.h"

#define OTM 0 /* optimal */
#define FAL 1 /* Fail */
#define OTW 2 /* optimal not waiting for without overflow */
#define OTO 3 /* optimal not waiting for with overflow */
#define SOW 4 /* sub-optimal without overflow */
#define SOO 5 /* sub-optimal with overflow */

/* inserir o código do Programa 14 aqui */

#endif
```

Programa 22: Arquivo system.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "system.h"
#include "knapsack.h"
#include "util.h"

CSystem::CSystem()
{
    m_iargc = 0;
    m_ppcargv = NULL;
    strcpy(sExt[0][0],""    );strcpy(sExt[0][1],""    );
    strcpy(sExt[0][2],""    );strcpy(sExt[0][3],""    );
    strcpy(sExt[1][0],""    );strcpy(sExt[1][1],".bkt1");
```

```
    strcpy(sExt[1][2],".bkt2");strcpy(sExt[1][3],""      );
    strcpy(sExt[2][0],""      );strcpy(sExt[2][1],".dyn1");
    strcpy(sExt[2][2],".dyn2");strcpy(sExt[2][3],""      );
    strcpy(sExt[3][0],""      );strcpy(sExt[3][1],".grd1");
    strcpy(sExt[3][2],".grd2");strcpy(sExt[3][3],".grd3");
}

CSystem::~CSystem(){}

void CSystem::SetArgs(int argc,char** argv)
{
    m_iargc   = argc;
    m_ppcargv = argv;
}

void CSystem::CompareSolutions(void)
{
    if (m_poKnapSackApproach->IsSolution())
    { /* Save Solution for approach */
      sprintf(sFileOut,"dset%1dn%03dM%03di%03d%s"
        ,dt,n,(int)dPM,inst,sExt[m_iOpt1][m_iOpt2]);
      sprintf(sTmp,"%s%s",sPathData,sFileOut);
      m_poKnapSackApproach->FileSaveSolution(sTmp);
    }

    if (m_poKnapSackApproach->IsOptimal())
    {
      if (!m_poKnapSackApproach->IsOverflow() && m_poKnapSackApproach->IsSolution())
      {
        if (m_poKnapSackApproach->Getfp() == m_poKnapSackOptimal->Getfp())
        {
          kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =OIM;
          time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
            m_poKnapSackApproach->GetTime();
          spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
            m_poKnapSackApproach->GetSpace();
          num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OIM]++;
          stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OIM] +=
            m_poKnapSackApproach->GetTime();
          sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OIM] +=
            m_poKnapSackApproach->GetSpace();
        }
        else if ( (m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
                  (m_poKnapSackOptimal->IsOptimal()) )
        {
          fprintf(pfOut
            ,"\nsolution found is better than optimal saved!(1)");
          m_poKnapSackApproach->ShowDatanM(pfOut);
          m_poKnapSackApproach->ShowDataSolution(pfOut);
          m_poKnapSackApproach->PrintState(pfOut);
          exit(0);
        }
        else if ( ( m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
                  (!m_poKnapSackOptimal->IsOptimal()) )
        {
          sprintf(sFileOut,"dset%1dn%03dM%03di%03d.opt"
            ,dt,n,(int)dPM,inst);
          sprintf(sTmp,"%s%s",sPathData,sFileOut);
```

```
            m_poKnapSackApproach->FileSaveSolution(sTmp);
          }
          else if ( (m_poKnapSackApproach->Getfp() < m_poKnapSackOptimal->Getfp()) )
          {
            fprintf(pfOut
              ,"\nimplementation internal error!(1)\n");
            m_poKnapSackApproach->PrintState(pfOut);
            m_poKnapSackApproach->ShowDatanM(pfOut);
            m_poKnapSackApproach->ShowDataSolution(pfOut);
            m_poKnapSackOptimal->PrintState(pfOut);
          m_poKnapSackOptimal->ShowDatanM(pfOut);
          m_poKnapSackOptimal->ShowDataSolution(pfOut);
          exit(0);
        }
      }
      else
      {
        fprintf(pfOut
          ,"\nimplementation internal error!(2)\n");
        m_poKnapSackApproach->PrintState(pfOut);
        exit(0);
      }
    }
    else
    { /* nao eh otimo */
      if ( (!m_poKnapSackApproach->IsSolution()) &&
           ( m_poKnapSackApproach->IsOverflow()) )
      { /* Fail */
        kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = FAL;
        num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][FAL]++;
      }
      else if ( (!m_poKnapSackApproach->IsSolution()) &&
                (!m_poKnapSackApproach->IsOverflow()) )
      { /* naum eh otimo, naum eh solucao e nao ocorreu estouro
          impossivel */
        fprintf(pfOut
          ,"\nimplementation internal error!(3)\n");
        m_poKnapSackApproach->PrintState(pfOut);
        exit(0);
      }
      else if (m_poKnapSackApproach->IsSolution() &&
               !m_poKnapSackApproach->IsOverflow())
      { /* nao eh otimo, eh Solucao e nao ocorreu estouro */
        if ( (m_poKnapSackApproach->Getfp() == m_poKnapSackOptimal->Getfp()) )
        {
          kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = OTW;
          time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
            m_poKnapSackApproach->GetTime();
          spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
            m_poKnapSackApproach->GetSpace();
          num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTW]++;
          stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTW] +=
            m_poKnapSackApproach->GetTime();
          sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTW] +=
            m_poKnapSackApproach->GetSpace();
        }
        else if ( (m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
                  (m_poKnapSackOptimal->IsOptimal()) )
```

```
    {
      fprintf(pfOut
        ,"solution found is better than optimal saved!(2)\n");
      m_poKnapSackApproach->ShowData(pfOut);
      m_poKnapSackApproach->ShowDataSolution(pfOut);
      m_poKnapSackApproach->PrintState(pfOut);
      exit(0);
    }
    else if ( ( m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
              (!m_poKnapSackOptimal->IsOptimal()) )
    { /* solucao nao-otima encontrada eh melhor que a anterior
         que tambem nao eh uma solucao otima */
      /* solucao pode ser otima, mas ... */
      kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =SOW;
      time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
        m_poKnapSackApproach->GetTime();
      spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
        m_poKnapSackApproach->GetSpace();
      num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW]++;
      stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        m_poKnapSackApproach->GetTime();
      sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        m_poKnapSackApproach->GetSpace();
      ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        (double)(m_poKnapSackOptimal->Getfp()-m_poKnapSackApproach->Getfp())/
                 m_poKnapSackOptimal->Getfp();
      /* saving to file, new solution not optimal */
      sprintf(sFileOut,"dset%1dn%03dM%03di%03d.opt"
        ,dt,n,(int)dPM,inst);
      sprintf(sTmp,"%s%s",sPathData,sFileOut);
      m_poKnapSackApproach->FileSaveSolution(sTmp);
    }
    else if ( (m_poKnapSackApproach->Getfp() < m_poKnapSackOptimal->Getfp()) )
    { /* solucao nao-otima encontrada eh pior que a anterior
          que nao eh uma solucao otima */
      kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =SOW;
      time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
        m_poKnapSackApproach->GetTime();
      spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
        m_poKnapSackApproach->GetSpace();
      num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW]++;
      stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        m_poKnapSackApproach->GetTime();
      sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        m_poKnapSackApproach->GetSpace();
      ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOW] +=
        (double)(m_poKnapSackOptimal->Getfp()-m_poKnapSackApproach->Getfp())/
                 m_poKnapSackOptimal->Getfp();
    }
  }
  else if (m_poKnapSackApproach->IsSolution() &&
           m_poKnapSackApproach->IsOverflow())
  { /* nao eh otimo, eh Solucao e ocorreu estouro */
    if (m_poKnapSackApproach->Getfp() == m_poKnapSackOptimal->Getfp())
    {
      kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =OTO;
      time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = 0;
      spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = 0;
```

```
    num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTO]++;
    stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTO] +=
      m_poKnapSackApproach->GetTime();
    sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][OTO] +=
      m_poKnapSackApproach->GetSpace();
  }
  else if ( (m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
            (m_poKnapSackOptimal->IsOptimal()) )
  {
    fprintf(pfOut
      ,"solution found is better than optimal saved!(3)\n");
    m_poKnapSackApproach->ShowData(pfOut);
    m_poKnapSackApproach->ShowDataSolution(pfOut);
    m_poKnapSackApproach->PrintState(pfOut);
    exit(0);
  }
  else if ( ( m_poKnapSackApproach->Getfp() > m_poKnapSackOptimal->Getfp()) &&
            (!m_poKnapSackOptimal->IsOptimal()) )
  { /* solucao nao-otima encontrada eh melhor que a anterior
      que tambem nao eh uma solucao otima */
    /* solucao pode ser otima, mas ... */
    kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = SOO;
    time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
      m_poKnapSackApproach->GetTime();
    spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
      m_poKnapSackApproach->GetSpace();
    num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO]++;
    stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      m_poKnapSackApproach->GetTime();
    sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      m_poKnapSackApproach->GetSpace();
    ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      (double)(m_poKnapSackOptimal->Getfp()-m_poKnapSackApproach->Getfp())/
              m_poKnapSackOptimal->Getfp();

    /* saving to file, new solution not optimal */
    sprintf(sFileOut,"dset%1dn%03dM%03di%03d.opt"
      ,dt,n,(int)dPM,inst);
    sprintf(sTmp,"%s%s",sPathData,sFileOut);
    m_poKnapSackApproach->FileSaveSolution(sTmp);
  }
  else if ( (m_poKnapSackApproach->Getfp() < m_poKnapSackOptimal->Getfp()) )
  { /* solucao nao-otima encontrada eh pior que a anterior
      que nao eh uma solucao otima */
    kind[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] = SOO;
    time[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
      m_poKnapSackApproach->GetTime();
    spc [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][inst-1] =
      m_poKnapSackApproach->GetSpace();
    num [dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO]++;
    stem[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      m_poKnapSackApproach->GetTime();
    sesp[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      m_poKnapSackApproach->GetSpace();
    ssot[dt-1][(int)(5*(b-1)+(n/pow(10,b)-1))][(int)dPM/10-1][SOO] +=
      (double)(m_poKnapSackOptimal->Getfp()-m_poKnapSackApproach->Getfp())/
              m_poKnapSackOptimal->Getfp();
  }
```

```
    }
  }
}

/* inserir o código do Programa 15 aqui */

/* inserir o código do Programa 16 aqui */

/* inserir o código do Programa 5 aqui */

/* inserir o código do Programa 17 aqui */
```

Programa 23: Arquivo system.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>

#include "system.h"

int main(int argc,char** argv)
{
  CSystem oSystem;

  if (    (argc < 2) ||
      !( ((argc == 5) && (strcmp(argv[1],"file"      )==0)) ||
         ((argc == 9) && (strcmp(argv[1],"batch"     )==0)) ||
         ((argc == 7) && (strcmp(argv[1],"generator")==0)) ||
         ((argc == 5) && (strcmp(argv[1],"resume"    )==0)) ) )
  {
    fprintf(stderr,"error, not enough parameters\n");
    fprintf(stderr,
      "usage: %s file      option1 option2 filedata\n",argv[0]);
    fprintf(stderr,
      "usage: %s batch     option1 option2 pathdata ds-df ns-nf
                           Ms-Mf insts-instf\n",argv[0]);
    fprintf(stderr,
      "usage: %s generator pathdata ds-df ns-nf
                           Ms-Mf insts-instf\n",argv[0]);
    fprintf(stderr,
      "usage: %s resume    pathdata ds-df Ms-Mf \n",argv[0]);
    return 1;
  }

  oSystem.SetArgs(argc,argv);

       if ( (argc == 5) && (strcmp(argv[1],"file"      )==0) )
  { /* file */
    return oSystem.ModuleFile();
  }
  else if ( (argc == 9) && (strcmp(argv[1],"batch"     )==0) )
  { /* batch */
    return oSystem.ModuleBatch();
  }
  else if ( (argc == 7) && (strcmp(argv[1],"generator")==0) )
```

```
  { /* generator */
    return oSystem.ModuleGenerator();
  }
  else if ( (argc == 5) && (strcmp(argv[1],"resume"    )==0) )
  { /* resume */
    return oSystem.ModuleResume();
  }
}
```

Programa 24: Arquivo main.cpp

```
#ifndef _STACKTIME_H_
#define _STACKTIME_H_

#include <time.h>

class CTime
{
public:
  clock_t tempo;;
  CTime* pNext;

  CTime()
  {
    tempo = clock();
    pNext = NULL;
  };
  CTime(clock_t _tempo)
  {
    tempo = _tempo;
    pNext = NULL;
  };
};

class CStackTime
{
private:
  CTime* pFirst;
  clock_t tStart,tFinish;
  double dElapTime;

  int      iCt;
public:
  CStackTime()
  {
    pFirst = NULL;
    iCt = 0;
  };

  int GetSize()
  {
    return iCt;
  };

  double GetElapTime()
  {
    return dElapTime;
  };
```

```
bool Push(CTime pt)
{
  CTime* ptAux;
  if ( (ptAux = new CTime(pt.tempo)) == NULL)
    return false;

  iCt++;

  if (pFirst != NULL)
    ptAux->pNext = pFirst;
  else
    ptAux->pNext = NULL;
  pFirst = ptAux;

  return true;
};

bool Pop (CTime& pt)
{
  CTime *ptAux;

  if (pFirst ==NULL)
    return false;

  pt.tempo = pFirst->tempo;

  ptAux = pFirst;
  pFirst = pFirst->pNext;
  delete ptAux;

  iCt--;

  return true;
};

void RemoveAll()
{
  CTime *ptAux;
  for(ptAux = pFirst;ptAux != NULL;ptAux = ptAux->pNext,iCt--)
    delete ptAux;
};

void Start()
{
  CTime oTempo;
  Push(oTempo);
}

void Finish()
{
  CTime oTimeFinish,oTimeStart;

  Pop(oTimeStart);

  tFinish = oTimeFinish.tempo;
  tStart  = oTimeStart.tempo;
```

```
    dElapTime = (double)(tFinish - tStart)/CLOCKS_PER_SEC;
  }

  char* GetStringTime(char* sTempo)
  {
    sprintf(sTempo,"[%3d:%02d:%02d.%03d]"
      ,(int)((tFinish-tStart)/CLOCKS_PER_SEC/3600)
      ,(int)((tFinish-tStart)/CLOCKS_PER_SEC/60   )%60
      ,(int)((tFinish-tStart)/CLOCKS_PER_SEC/1    )%60
      ,(int)(1000.*(tFinish-tStart)/CLOCKS_PER_SEC)%1000);
    return sTempo;
  }
};

#endif
```

Programa 25: Arquivo stacktime.h

```
#ifndef _HEAPSORT_H_
#define _HEAPSORT_H_

#include "tda.h"

void Heapsort(Object *A, Index *n,int bAsc);

#endif
```

Programa 26: Arquivo heapsort.h

```
#include<limits.h>
#include <stdio.h>

#include "heapsort.h"

void ReDo(Index Left, Index Right, Object *A,int bAsc)
{
  Index i = Left;
  int j;
  Object x;
  j = i * 2;
  x = A[i];
  while (j <= Right)
  {
    if (j < Right)
    {
      if ( ( (A[j].Key < A[j+1].Key) && ( bAsc) ) ||      /* here */
           ( (A[j].Key > A[j+1].Key) && (!bAsc) ) )
        j++;
    }
    if ( ( (x.Key >= A[j].Key) && ( bAsc) ) ||
         ( (x.Key <= A[j].Key) && (!bAsc) ) )
      goto L999; /* aqui */
    A[i] = A[j];
    i = j; j = i * 2;
  }
L999:
  A[i] = x;
}
```

```
void BuildUp(Object *A, Index *n,int bAsc)
{
  Index Left;
  Left = *n / 2 + 1;
  while (Left > 1)
  {
    Left--;
    ReDo(Left, *n, A, bAsc);
  }
}

void Heapsort(Object *A, Index *n,int bAsc)
{
  Index Left, Right;
  Object x;
  BuildUp(A, n, bAsc); /* BuildUp Heap */
  Left = 1; Right = *n;
  while (Right > 1)
  {
    /* ordena o vetor */
    x = A[1]; A[1] = A[Right]; A[Right] = x;
    Right--;
    ReDo(Left, Right, A, bAsc);
  }
}
```

Programa 27: Arquivo heapsort.cpp

```
#ifndef _UTIL_H_
#define _UTIL_H_

#define TRUE 1
#define FALSE 0

#define max(x,y) ( ((x) > (y)) ? (x) : (y) )

double uniformrand0to1();
void   parameter2(char*,int&,int&);

#endif
```

Programa 28: Arquivo util.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>

#include "util.h"

/* inserir o código do Programa 3 aqui */

void parameter2(char* s,int& ps,int& pf)
{
  char tmp[BUFSIZ],*p;
```

```
  strcpy(tmp,s);
  p=strchr(tmp,'-');
  if (p==NULL)
  {
    ps = atoi(tmp);
    pf = ps;
  }
  else
  {
    tmp[p-tmp] = '\0';
    pf = atoi(++p);
    ps = atoi(tmp);
  }
}
```

Programa 29: Arquivo util.cpp

Referências

[1] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, 1978.

[2] Bernard M. E. Moret and Henry D. Shapiro. *Algorithms from P to NP*, volume I. The Benjamin/Cummings Publishing Company, Inc., 1990.

[3] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[4] Robert Sedgewick. *Algorithms in C.* Addison-Wesley, 1990.

[5] Nivio Ziviani. *Projeto de Algoritmos com implementações em Pascal e C.* Pioneira Thomson Learnig, São Paulo - SP - Brasil, 2a. edition, 2004. revisada e ampliada.