Introdução à construção de heurísticas baseadas na formulação matemática do problema utilizando ferramenta de solução matemática comercial

Marcos Mansano Furlan

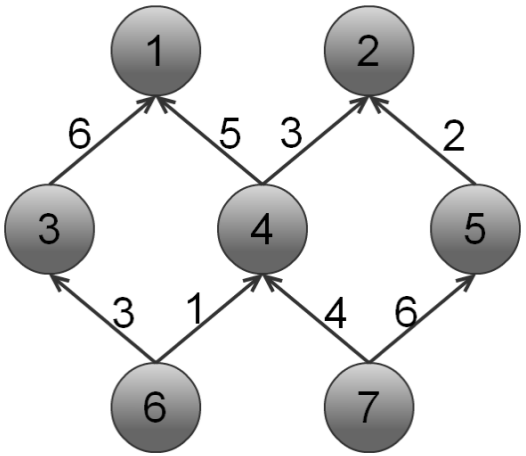
Faculdade de Ciências Exatas e Tecnologias - UFGD

Agenda

- Problema abordado (MLCLSP)
- Estruturas de dados
- Funções utilizadas
- Heurísticas implementadas
- Fontes e referências bibliográficas

Problema abordado (MLCLSP)

- Dimensionamento de lotes;
- Estrutura de produtos com múltiplos níveis;
- Demanda externa dos itens finais;
- Recursos limitados.



MLCLSP

$$\text{minimize } Z = \sum_{i=1}^{N}\sum_{t=1}^{T}(s_i * Y_{it} + h_i * I_{it}) + \sum_{m=1}^{M}\sum_{t=1}^{T} oc_m * O_{mt} \quad (1)$$

$$\text{sujeito a: } I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} * X_{j,t} = d_{it} + I_{it} \quad \forall i, t \quad (2)$$

$$\sum_{i \in K_m} (a_i * X_{it} + ts_i * Y_{it}) \leq C_{mt} + O_{mt} \ \forall m, t \quad (3)$$

$$X_{it} \leq B_{it} * Y_{it} \quad \forall i, t \quad (4)$$

$$X_{it}, I_{it} \geq 0 \quad \forall i, t \quad (5)$$

$$Y_{it} \in \{0, 1\} \quad \forall i, t \quad (6)$$

Estruturas de dados utilizadas

- Dados
  - Reais
    - `IloNum, IloNumArray, IloArray<IloNumArray>`
  - Inteiros
    - `IloInt, IloIntArray, IloArray<IloIntArray>`
- Variáveis
  - Reais
    - `IloNumVar, IloNumVarArray, IloArray<IloNumVarArray>`
  - Inteiros
    - `IloIntVar, IloIntVarArray, IloArray<IloIntVarArray>`

Estruturas de dados utilizadas

- Demanda (d_{it})
  - `IloArray<IloIntArray> demanda(env,N);`
  - `for(int i=0;i<N;i++)`
    - `demanda[i] = IloIntArray(env,T);`
- Tamanho dos lotes (X_{it})
  - `IloArray<IloNumVarArray> X(env,N);`
  - `for(int i=0;i<N;i++)`
    - `X[i] = IloNumVarArray(env,T, 0,IloInfinity, ILOFLOAT);`

Estruturas de dados utilizadas

- Preparaação de máquina (Y_{it})
  - `IloArray<IloNumVarArray> Y(env,N);`
  - `for(int i=0;i<N;i++)`
    - `Y[i] = IloNumVarArray(env,T, 0,1, ILOINT);`

Estruturas de dados utilizadas

- Conjunto de números inteiros
  - `IloIntSet A(env);`
- Funções
  - Adição de valor
    - `A.add(valor);`
  - Remoção de valor
    - `A.remove(valor);`
  - Verificar se o valor pertence ao conjunto
    - `A.contains(valor);`

Estruturas de dados utilizadas

- Iterador de conjunto de números inteiros
  - `IloIntSet::Iterator iter(A);`
- Descrição
  - Elemento utilizado para percorrer os elementos de um conjunto
  - Função ok (utilizada para verificar se o iterador esta apontando para um elemento do conjunto
    - `for(IloIntSet::Iterator ite(S[i]);ite.ok();++ite)`

MLCLSP

$$\text{minimize } Z = \sum_{i=1}^{N}\sum_{t=1}^{T}(s_i * Y_{it} + h_i * I_{it}) + \sum_{m=1}^{M}\sum_{t=1}^{T} oc_m * O_{mt} \qquad (1)$$

```
//Inicio da criaÁ„o da funÁ„o objetivo
obj.clear();
for(i=1;N.contains(i);i++)
    for(t=1;T.contains(t);t++)
        obj+=s[i]*(*y)[i][t]+h[i]*I[i][t];
for(m=1;M.contains(m);m++)
    for(t=1;T.contains(t);t++)
        obj+= oc[m]*O[m][t];
agregado->add(IloMinimize(env, obj));
//FunÁ„o Objetivo adicionada
```

MLCLSP (caso 1 – t=2,...T-1)

$$I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} * X_{j,t} = d_{it} + I_{it} \quad \forall i, t \qquad (2)$$

```
restr.clear();
for(i=1;N.contains(i);i++){
    for(t=2;t!=T.getLast();t++){
        restr+=I[i][t-1]+x[i][t]-I[i][t];
        for(IloIntSet::Iterator ite(S[i]);ite.ok();++ite){
            restr-=r[i][*ite]*x[*ite][t+(int)lt[i]];
        }
        agregado->add(restr==d[i][t]);
        restr.clear();
    }
}
```

MLCLSP (caso 2 - t=T)

$$I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} * X_{j,t} = d_{it} + I_{it} \quad \forall i, t \qquad (2)$$

```
restr.clear();
for(i=1;N.contains(i);i++){
    restr+=I[i][T.getLast()-1]+x[i][T.getLast()]-I[i][T.getLast()];
    if(lt[i]==0)
        for(IloIntSet::Iterator ite(S[i]);ite.ok();++ite)
            restr-=r[i][*ite]*x[*ite][t+(int)lt[i]];
    agregado->add(restr==d[i][T.getLast()]);
    restr.clear();
}
```

MLCLSP (caso 3 – t=1)

$$I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} * X_{j,t} = d_{it} + I_{it} \quad \forall i, t \qquad (2)$$

```
restr.clear();
for(i=1;N.contains(i);i++){
    restr+=I0[i]+x[i][1]-I[i][1];
    for(IloIntSet::Iterator ite(S[i]);ite.ok();++ite){
        restr-=r[i][*ite]*x[*ite][1+(int)lt[i]];
    }
    agregado->add(restr==d[i][1]);
    restr.clear();
}
```

MLCLSP (caso 3 – t=1)

$$I_{i,t-1} + X_{it} - \sum_{j \in S(i)} r_{ij} * X_{j,t} = d_{it} + I_{it} \quad \forall i, t \qquad (2)$$

```
restr.clear();
for(i=1;N.contains(i);i++){
     restr+=Iinic[i]-I0[i];
     if(lt[i]!=0)
          for(IloIntSet::Iterator ite(S[i]);ite.ok();++ite){
               restr-=r[i][*ite]*x[*ite][(int)lt[i]];
          }
     agregado->add(restr==0);
     restr.clear();
}
```

MLCLSP

$$\sum_{i \in K_m} (a_i * X_{it} + ts_i * Y_{it}) \leq C_{mt} + O_{mt} \ \forall m, t \qquad (3)$$

```
restr.clear();
for(t=1;T.contains(t);t++)
    for(m=1;M.contains(m);m++){
        for(IloIntSet::Iterator ite(K[m]);ite.ok();++ite)
            restr+=a[*ite]*x[*ite][t]+ts[*ite]*(*y)[*ite][t];
        agregado->add(restr<=(C[m][t]+O[m][t]));
        restr.clear();
    }
```

MLCLSP

$$X_{it} \leq B_{it} * Y_{it} \quad \forall i, t \qquad (4)$$

```
for(t=1;T.contains(t);t++){
    for(i=1;N.contains(i);i++){
        agregado->add(x[i][t]<=D[i][t]*(*y)[i][t]);
    }
}
```

Funções utilizadas nas heurísticas

- Mudança de tipo
  - `IloConversion(env, Y[i][t], ILOFLOAT);`
  - `modelo.add(IloConversion(env, Y[i][t], ILOBOOL));`
- Descrição
  - Converter o tipo da variável
  - Duas conversões não podem ser adicionadas na mesma variável
    - `IloConversion(env, Y[1][1], ILOFLOAT);`
    - `IloConversion(env, Y[1][1], ILOBOOL);`
  - Uma conversão pode ser removida
    - `IloConversion conv(env, Y[1][1], ILOFLOAT);`
    - `modelo.add(conv);`
    - `...`
    - `modelo.remove(conv);`

Funções utilizadas nas heurísticas

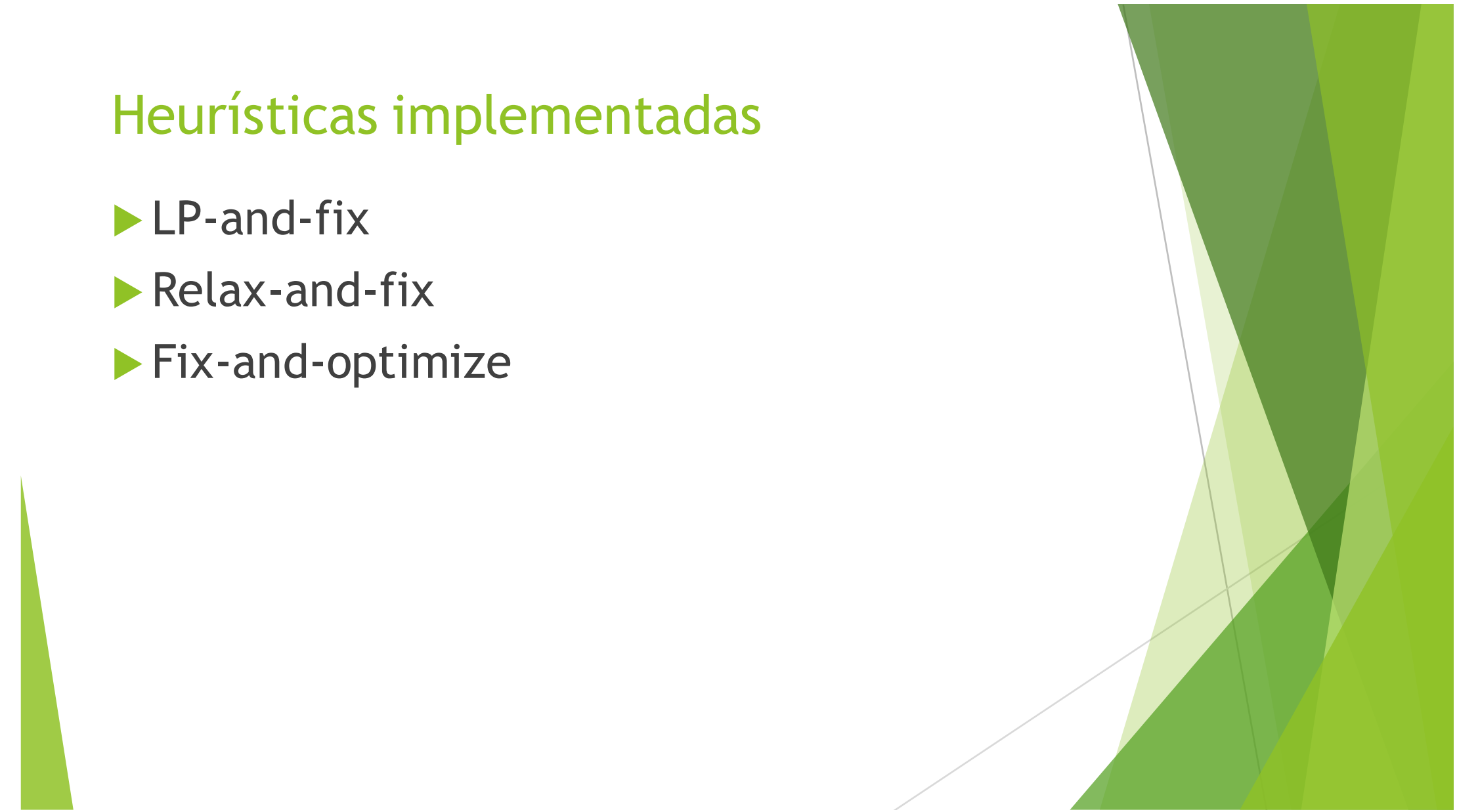
- Obter valor da variável
  - `modelo.getValue(y[i][t]);`
  - `modelo.getValue(y[i][t], soln);`
- Descrição
  - Obter o valor da variável na última execução do CPLEX
  - Sempre retorna valor de ponto flutuante
  - Gera erro se não houver solução

Funções utilizadas nas heurísticas

- Fixar valor de variável (mudar limitantes)
  - Y[i][t].setBounds(lb,ub);
- Descrição
  - Função utilizada para mudar os limitantes de uma variável
  - Pode ser utilizada para fixar o valor (lb=ub)
  - Evita o aumento do modelo com a inclusão de novas restrições

Heurísticas implementadas

- LP-and-fix
- Relax-and-fix
- Fix-and-optimize



Heurística LP-and-fix

- Heurística baseada na modelagem do problema;
- Ideia:
  - Reduzir o problema resultante através de fixação;
  - Baseado na relaxação linear completa do problema;
  - Melhores resultados para modelos com relaxação linear forte;
  - Inicialmente proposto para o problema de lot-sizing por Maes et al. (1991).

Heurística LP-and-fix

1. Resolver a relaxação linear do problema;
2. Fixar todas as variáveis que tenham resultado inteiro na relaxação;
3. Resolver o sub-MIP fixado por um método de solução, como por exemplo um *branch-and-bound* ou uma ferramenta como o Cplex.

Modificação: Considere p tal que 0 <= p < 1:

```
if(variavel >= 1 - p){
 fixe a variavel em 1;
}
```

Heurística LP-and-fix

```
IloModel modelo;
matrizvarnum y;
//Monta Modelo
MLCLSP(env, N, M, T, K, S, A, r, C, B, d, oc, h, s, a, ts, lt, Iinic, &modelo, &y);
//Relaxa as variáveis
IloExtractableArray relaxa(env);
for(int t=1;t<=T.getSize();t++){
    for(int i=1;i<=N.getSize();i++){
        relaxa.add(IloConversion(env,y[i][t],ILOFLOAT));
    }
}
```

Heurística LP-and-fix

```
//Adiciona e resolve o modelo relaxado
modelo.add(relaxa);
IloCplex cplex(modelo);
cplex.solve();
//Salvar solução
matriznum sol(env,N.getSize()+1);
for(int i=1;i<=N.getSize();i++){
    sol[i] = IloNumArray(env,T.getSize()+1);
    for(int t=1;t<=T.getSize();t++){
        sol[i][t] = cplex.getValue(y[i][t]);
    }
}
//Remove as relaxações
modelo.remove(relaxa);
```

Heurística LP-and-fix

```
int fixaZero=0, fixaUm=0;
for(int i=1;i<=N.getSize();i++){
    for(int t=1;t<=T.getSize();t++){
        if(sol[i][t]>=0.8){
            y[i][t].setBounds(1,1);
            fixaUm++;
        }else if(sol[i][t]<=0.02){
            y[i][t].setBounds(0,0);
            fixaZero++;
        }else{
            y[i][t].setBounds(0,1);
        }
    }
}
cout << "# var fixadas em zero "<< fixaZero <<endl;
cout << "# var fixadas em um "<< fixaUm <<endl;
cplex.setParam(IloCplex::TiLim,600);
cplex.solve();
```

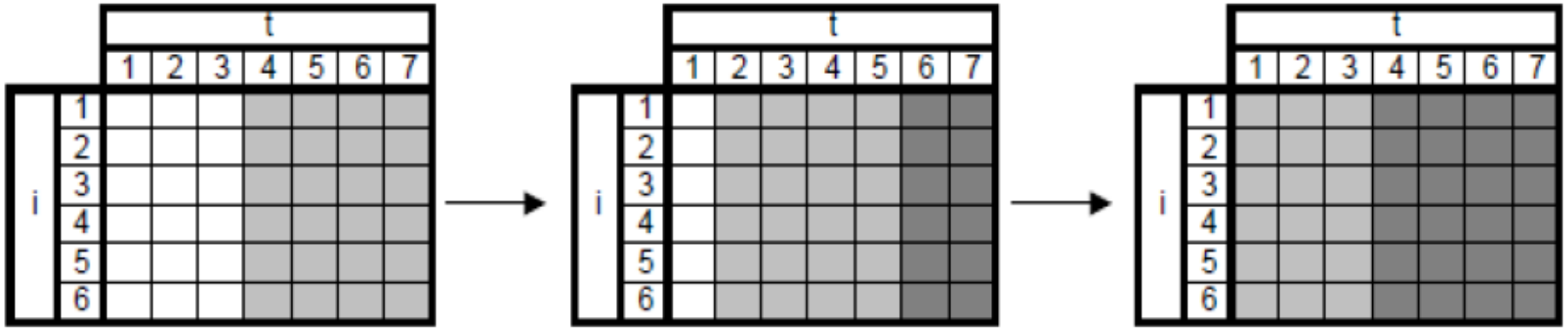
Heurística LP-and-fix

```
ATLANTIDA:201601ICMC mafurlan$ ./main ./classap/A_____0.set ./classap/AGC2_10.cap ./classap/AG
_2.dem ./classap/AG_23.sco ./classap/AGC.dat ./classap/AGC2310.res 1
Tried aggregator 1 time.
LP Presolve eliminated 13 rows and 14 columns.
Aggregator did 248 substitutions.
Reduced LP has 301 rows, 541 columns, and 1257 nonzeros.
Presolve time = 0.00 sec. (0.46 ticks)

Iteration log . . .
Iteration:     1   Dual objective     =          -592.528109
Iteration:    76   Dual objective     =         10141.480178
Iteration:   159   Dual objective     =         19033.165938
Iteration:   235   Dual objective     =         27906.228880
Iteration:   310   Dual objective     =         39635.787835
# var fixadas em zero 26
# var fixadas em um 4
Found incumbent of value 9.0117085e+08 after 0.00 sec. (1.93 ticks)
Probing fixed 6 vars, tightened 0 bounds.
Probing time = 0.00 sec. (0.08 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 0.00 sec. (0.90 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                       9.01171e+08     2560.0000       18  100.00%
      0     0    50051.0453   199   9.01171e+08    50051.0453       18   99.99%
*     0+    0                       202038.6935    50051.0453       18   75.23%
      0     0    72009.7933   170   202038.6935     Cuts: 240      258   64.36%
```

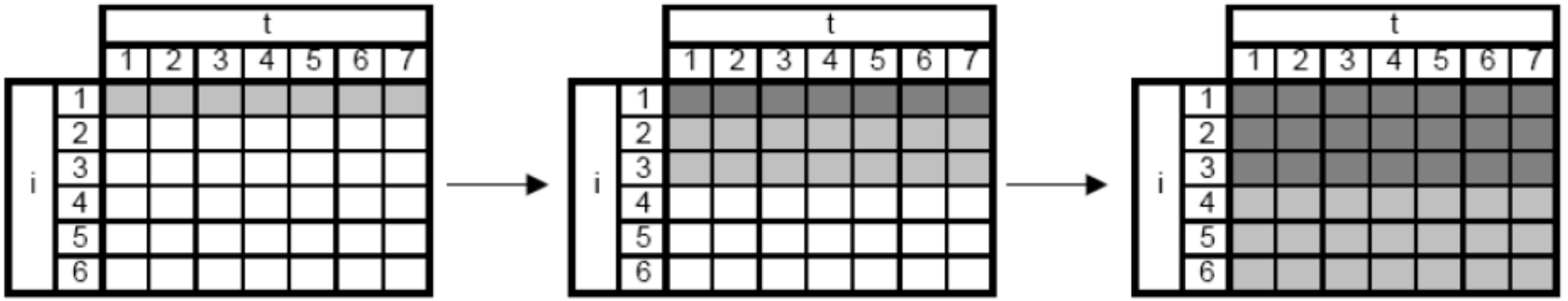
Relax-and-fix

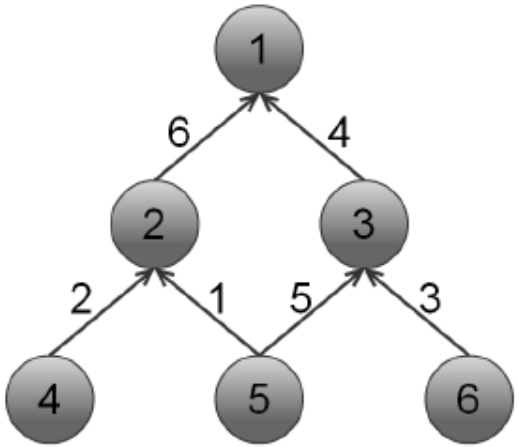
- Heurística baseada na modelagem do problema;
- Ideia:
  - Particionar as variáveis inteiras do problema em R sub-conjuntos tal que: Q(r), r=1..R são as partições;
  - Inicia-se a resolução com o problema relaxado;
  - A cada iteração, retornamos a integralidade de uma partição e resolvemos o sub-problema;
  - Fixa-se o valor das variáveis inteira e um novo sub-conjunto é escolhido para que suas variáveis se tornem inteiras.

Relax-and-fix



Relax-and-fix





Relax-and-fix

```
IloModel modelo;
matrizvarnum y;
//Monta Modelo
MLCLSP(env, N, M, T, K, S, A, r, C, B, d, oc, h, s, a, ts, lt, Iinic, &modelo, &y);
//Relaxa as variáveis
IloArray<IloExtractableArray> relaxa(env,N.getSize()+1);
for(int i=1;i<=N.getSize();i++)
    relaxa[i] = IloExtractableArray(env,T.getSize()+1);
for(int t=1;t<=T.getSize();t++){
    for(int i=1;i<=N.getSize();i++){
        relaxa[i][t] = IloConversion(env,y[i][t],ILOFLOAT);
    }
}
//Adiciona e resolve o modelo relaxado
for(int t=1;t<=T.getSize();t++)
    for(int i=1;i<=N.getSize();i++)
        modelo.add(relaxa[i][t]);
IloCplex cplex(modelo);
cplex.setOut(env.getNullStream());
IloNum objFO = IloInfinity;
//Salvar solução
matriznum sol(env,N.getSize()+1);
for(int i=1;i<=N.getSize();i++)
    sol[i] = IloNumArray(env,T.getSize()+1);
```

Relax-and-fix

```
//Relax-and-fix por períodos
for(int t=1;t<=T.getSize();t++){
    cout << "Iteracao " << t << tab << objFO << endl;
    //Remove as relaxações do período
    for(int i=1;i<=N.getSize();i++)
        modelo.remove(relaxa[i][t]);
    cplex.solve();
    objFO  = cplex.getObjValue();
    //Salva a parte inteira da solução
    for(int i=1;i<=N.getSize();i++)
        sol[i][t] = cplex.getValue(y[i][t]);
    //Fixa a parte inteira da solução
    for(int i=1;i<=N.getSize();i++){
        if(sol[i][t]>=0.8){
            y[i][t].setBounds(1,1);
        }else{
            y[i][t].setBounds(0,0);
        }
    }
}
cout << "Resultado Final " << objFO << endl;
```

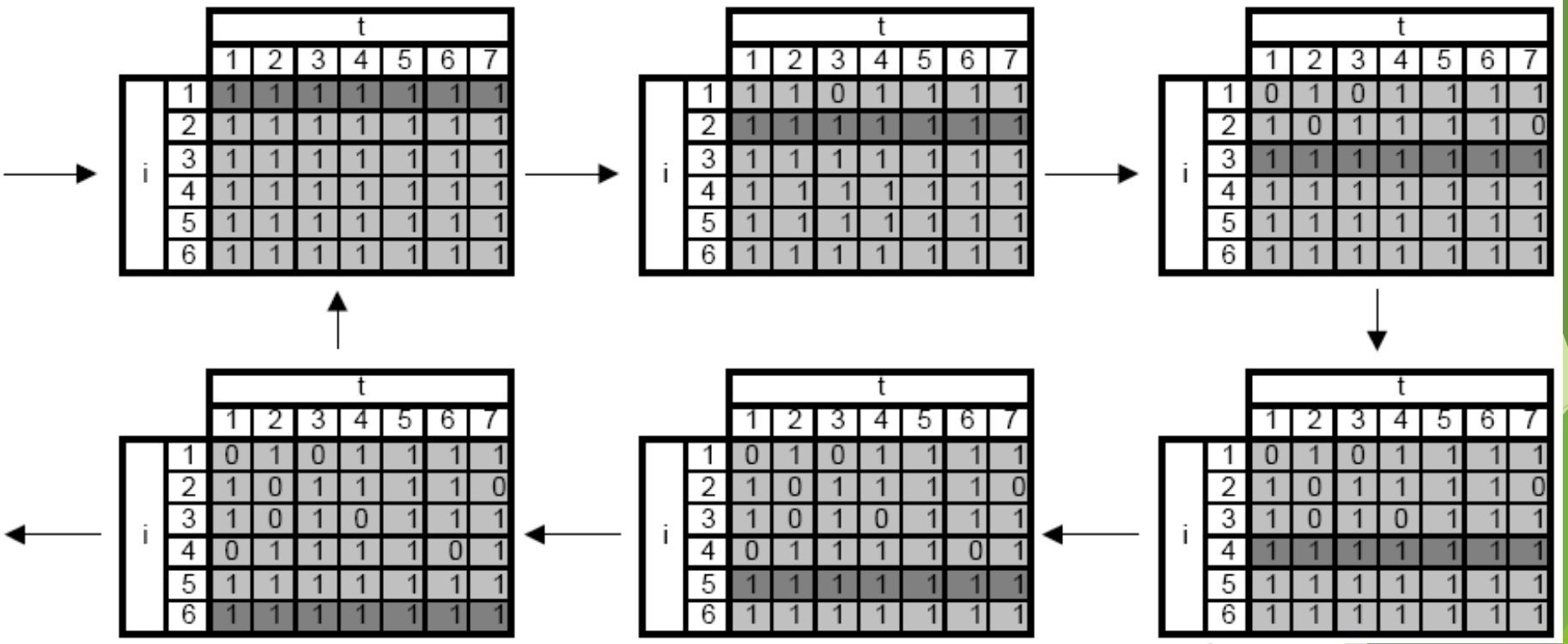
Relax-and-fix

```
ATLANTIDA:201601ICMC mafurlan$ ./main ./classap/A_____0.set ./classap/AGC2_10.cap ./classap/AG
_2.dem ./classap/AG_23.sco ./classap/AGC.dat ./classap/AGC2310.res 2
Iteracao 1      inf
Iteracao 2      49894.8
Iteracao 3      54690.4
Iteracao 4      59812.8
Iteracao 5      65342.9
Iteracao 6      71096.4
Iteracao 7      76990.8
Iteracao 8      83111.1
Iteracao 9      88135.7
Iteracao 10     91978
Iteracao 11     97102.3
Iteracao 12     102365
Iteracao 13     106649
Iteracao 14     112194
Iteracao 15     117516
Iteracao 16     122836
Iteracao 17     128777
Iteracao 18     134521
Iteracao 19     140086
Iteracao 20     144283
Iteracao 21     150604
Iteracao 22     154955
Iteracao 23     156773
Iteracao 24     157939
Resultado Final 158067
```

Fix-and-optimize

- Heurística fixe-e-otimize
- Pochet e Wolsey (2006): Exchange;
- Baseia-se na formulação matemática do problema;
- Particionamento das variáveis inteiras;
- Para cada Qr - um subproblema:
- Qr é liberado para otimização;
- As demais variáveis inteiras são fixadas;

Fix-and-optimize



Fix-and-optimize

```
IloModel modelo;
matrizvarnum y;
//Monta Modelo
MLCLSP(env, N, M, T, K, S, A, r, C, B, d, oc, h, s, a, ts, lt, Iinic, &modelo, &y);
IloCplex cplex(modelo);
cplex.setOut(env.getNullStream());
IloNum objFO = IloInfinity;
//Solução inicial lote-por-lote
matriznum sol(env,N.getSize()+1);
for(int i=1;i<=N.getSize();i++){
    sol[i] = IloNumArray(env,T.getSize()+1);
    for(int t=1;t<=T.getSize();t++)
        sol[i][t]=1;
}
```

Fix-and-optimize

```
//fix-and-opt por períodos
for(int t=1;t<=T.getSize();t++){
    cout << "Iteracao " << t << tab << objFO << endl;
    //Fixa a parte inteira da solução
    for(int t2=1;t2<=T.getSize();t2++){
        for(int i=1;i<=N.getSize();i++){
            if(t2!=t){
                if(sol[i][t2]>=0.8){
                    y[i][t2].setBounds(1,1);
                }else{
                    y[i][t2].setBounds(0,0);
                }
            }else{
                y[i][t2].setBounds(0,1);
            }
        }
    }
    cplex.solve();
    objFO  = cplex.getObjValue();
    for(int i=1;i<=N.getSize();i++)
        sol[i][t]=cplex.getValue(y[i][t]);

}
cout << "Resultado Final " << objFO << endl;
```

Fix-and-optimize

```
[ATLANTIDA:201601ICMC mafurlan$ ./main ./classap/A_____0.set ./classap/AGC2_10.cap ./classap/AG
_2.dem ./classap/AG_23.sco ./classap/AGC.dat ./classap/AGC2310.res 3
Iteracao 1      inf
Iteracao 2      220320
Iteracao 3      216693
Iteracao 4      213474
Iteracao 5      210741
Iteracao 6      208246
Iteracao 7      204606
Iteracao 8      200165
Iteracao 9      197311
Iteracao 10     193251
Iteracao 11     189423
Iteracao 12     186805
Iteracao 13     183661
Iteracao 14     180913
Iteracao 15     178769
Iteracao 16     178029
Iteracao 17     177451
Iteracao 18     175849
Iteracao 19     173806
Iteracao 20     172548
Iteracao 21     168790
Iteracao 22     168790
Iteracao 23     165809
Iteracao 24     162558
Resultado Final 157934
```

Fontes e referências bibliográficas

- Documentação da interface C++