Stevan Kordić & Nataša Kovač
Maritime Faculty – University of Montenegro

# *One Combinatorial Algorithm for Berth Allocation Problem*

*Fourth Workshop*
On Formal & Automated Theorem Proving & Applications
February 4-5, 2011, Belgrade, Serbia

# Input Data

***T***: Total number of time periods in the planning horizon.

***m*** : The number of berths in the port.

***l*** : The number of vessels in the planning horizon.

***vessel:*** Sequence of data relevant for vessels, it has following structure:

$$vessel = \{ (ETA_k, a_k, b_k, d_k, s_k, c_{1k}, c_{2k}, c_{3k}, c_{4k}) \mid k = 1...l \}.$$

Members of the 9-tuple represents following data of the vessel:

$ETA_k$: Expected time of arrival of $vessel_k$;

$a_k$: The processing time of $vessel_k$ if only one crane is assigned to $vessel_k$;

$b_k$: The length of $vessel_k$;

$d_k$: The due time for the departure of $vessel_k$;

$s_k$: The least-cost berthing location of the reference point of $vessel_k$;

$c_{1k}$: The penalty cost of $vessel_k$ if the vessel could not dock at its preferred berth;

$c_{2k}$: The penalty cost of $vessel_k$ per unit time of earlier arrival before $ETA_k$;

$c_{3k}$: The penalty cost of $vessel_k$ per unit time of late arrival after $ETA_k$;

$c_{4k}$: The penalty cost of $vessel_k$ per unit time delay behind the due time $d_k$.

## 3.1.2 Decision variables and domains

$At_k$ **:** The arrival time of vessel *k* to the berth.

Domain $(At_k)=\{1,2,3,4,\ldots,T\}$

$Dt_k$ **:** The departing time of vessel *k*.

Domain $(Dt_k)=\{1,2,3,4,\ldots,T\}$

$X_{itk}$ **:** 1 if the berth i at time t is allocated to vessel k, otherwise 0.

Domain $(X_{itk})=\{0,1\}$

### 3.1.3 Constraints

**Constraint 3-1-1:** The grid squares are covered by only one vessel. In fact, each berth at time t can be assigned to only one vessel.

$$\sum_{k=1}^{l} X_{itk} \leq 1 \ \ for\ i = 1,2,3,...,m; t = 1,2,3,..,T$$

**Constraint 3-1-2:** Each berth is allocated for the vessel only between its arrival and departure.

$$At_k \leq t \leq Dt_k \Rightarrow X_{itk} = 1$$
$$(At_k \succ t) \ \ OR\ (t \succ Dt_k) \Rightarrow X_{itk} = 0$$
$$for\ t = 1,2,..,T; for\ i = 1,2,..,m; for\ k = 1,2,..,l$$

### 3.1.4 Objective function

The objective function of this decision is to minimize the total penalty costs. In order to present the objective function, we introduce the following auxiliary variable:

$Z_k$ : The sum of the absolute distance between the preferred location of vessel k and the berths allocated to the vessel. This variable is determined by the following equation:

$$Z_k = f(X_{itk}, s_k) = \sum_{t=1}^{T} \sum_{i=1}^{m} \left\{ i - s_k \middle| : X_{itk} = 1 \right\}$$

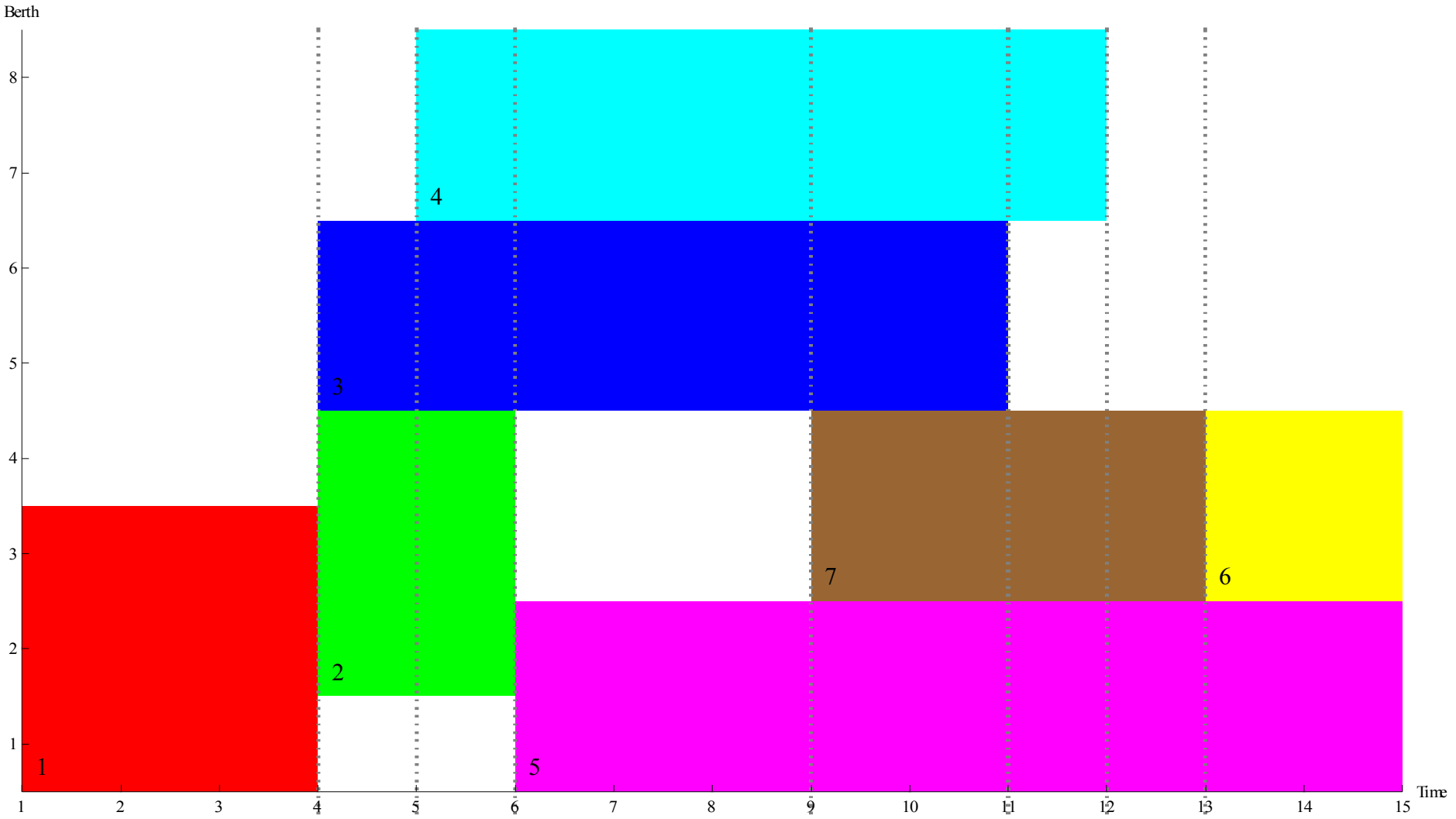
Now the objective function is written as follows:

$$MinCostVessels = \sum_{k=1}^{l} \left\{ c_{1k}.Z_k + c_{2k}(ETA_k - At_k)^+ + c_{3k}(At_k - ETA_k)^+ + c_{4k}(Dt_k - d_k)^+ \right\}$$

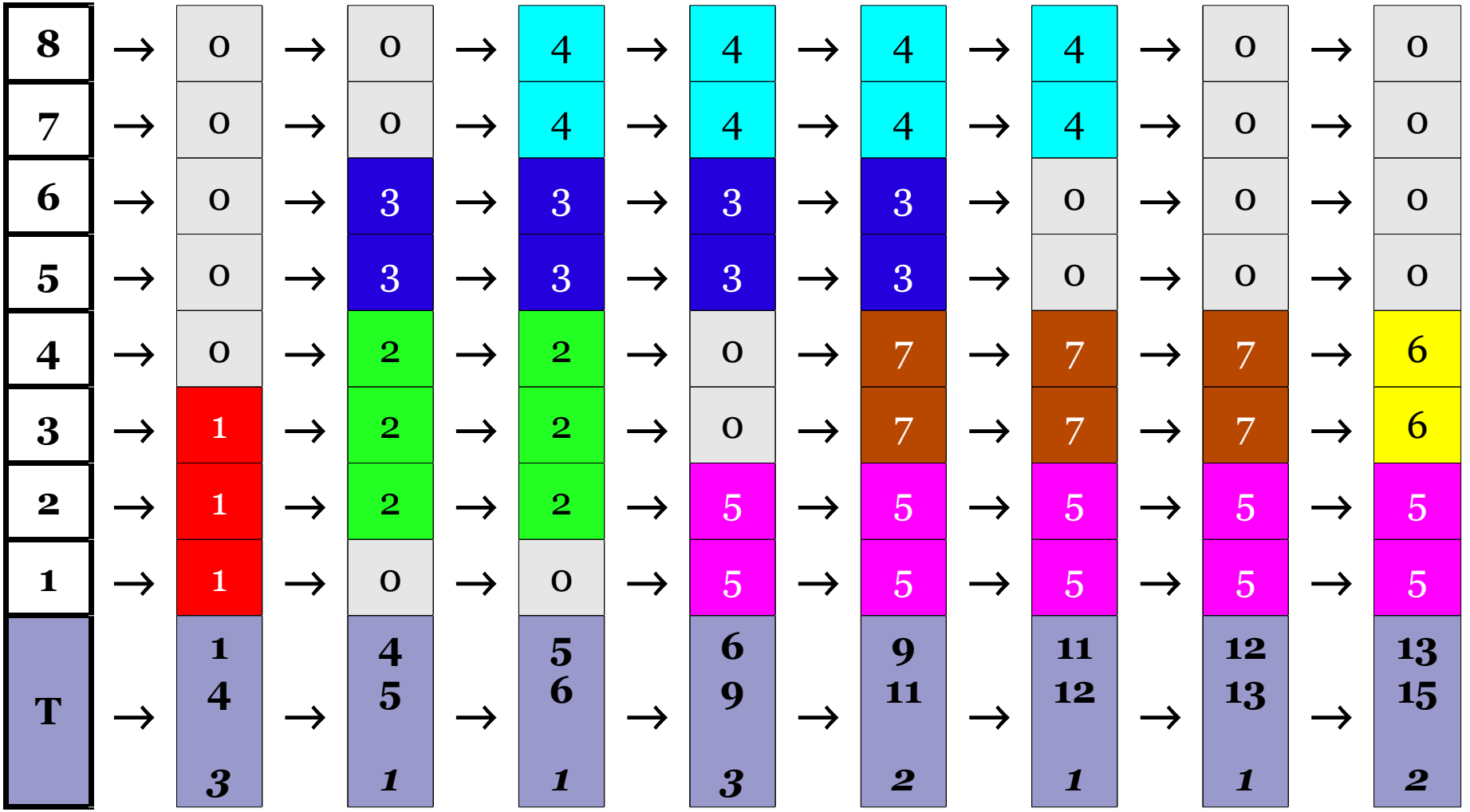
# Example of input data

| $k$ | $ETA_k$ | $a_k$ | $b_k$ | $d_k$ | $s_k$ | $c_{1k}$ | $c_{2k}$ | $c_{3k}$ | $c_{4k}$ |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 9 | 3 | 4 | 1 | 25 | 10 | 10 | 30 |
| **2** | 4 | 6 | 3 | 6 | 3 | 40 | 30 | 20 | 35 |
| **3** | 4 | 14 | 2 | 11 | 5 | 10 | 20 | 20 | 25 |
| **4** | 5 | 14 | 2 | 12 | 7 | 15 | 20 | 20 | 30 |
| **5** | 6 | 18 | 2 | 16 | 2 | 35 | 25 | 20 | 45 |
| **6** | 13 | 4 | 2 | 16 | 4 | 20 | 30 | 35 | 40 |
| **7** | 10 | 8 | 2 | 14 | 3 | 25 | 10 | 10 | 30 |
| **8** | 6 | 6 | 2 | 9 | 4 | 40 | 35 | 40 | 70 |
| **9** | 2 | 8 | 4 | 2 | 2 | 20 | 30 | 30 | 25 |

# Example of output data



# Internal list data structure representation



# Distribution of cost for a vessel with ETA=***3*** at berth **5**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **m** | | | | | | | . . . | | |
| **m-1** | | | | | | | . . . | | |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| **7** | 50 | 40 | 30 | 40 | 50 | 60 | . . . | | |
| **6** | 40 | 30 | 20 | 30 | 40 | 50 | . . . | | |
| **5** | 30 | 20 | **10** | 20 | 30 | 40 | . . . | | |
| **4** | 40 | 30 | 20 | 30 | 40 | 50 | . . . | | |
| **3** | 50 | 40 | 30 | 40 | 50 | 60 | . . . | | |
| **2** | 60 | 50 | 40 | 50 | 60 | 70 | . . . | | |
| **1** | 70 | 60 | 50 | 60 | 70 | 80 | . . . | | |
| Berth *Time* | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***. . .*** | ***T-1*** | ***T*** |

Sequence **ξ** for a vessel with ETA=***4*** at berth **5**

$$\xi = \{ (5,3,10),$$
$$(4,3,20), (5,2,20), (5,4,20), (6,3,20),$$
$$(3,3,30), (4,2,30), (4,4,30), \ldots (7,3,30),$$
$$\vdots$$
$$(m,T,x) \}$$

# *Brute Force Algoritam*

```
MinCost = +∞;
n  = Table[0, {ID,l}];
ID = 1;
λ  = 1;
stack = ξ;
While 0 < ID Do
          DeleteVessel[ID];
          If λ == -1 Then
               n[ID] = 0; ID--;
               Pop[];
               λ = 1,
          Else
               If n[ID] < Length[ξ[ID]] Then
                    n[ID]++;
                    berth     = ξ[ID,n[ID],1];
                    arrival   = ξ[ID,n[ID],2];
                    tempCost  = Sum[ξ[i,n[i],3],{i,ID}];
                    InsertVessel[ID,berth,arrival];
                    If ID == l Then
                         ReportSolution[tempCost];
                         λ = -1
                    Else
                         If tempCost < minCost Then
                              Push[];
                              ID++
                         Else
                              λ = -1
                         EndIf
                    EndIf
               Else
                    λ = -1
               EndIf
          EndIf
EndWhile
```

# *Algoritam Opt 1*

```
MinCost = +∞;
n  = Table[0, {ID,l}];
ID = 1;
λ  = 1;
stack = ξ;
While 0 < ID Do
          DeleteVessel[ID];
          If λ == -1 Then
               n[ID] = 0; ID--;
               Pop[];
               λ = 1,
          Else
               If n[ID] < Length[ξ[ID]] Then
                    n[ID]++;
                    berth     = ξ[ID,n[ID],1];
                    arrival   = ξ[ID,n[ID],2];
                    tempCost  = Sum[ξ[i,n[i],3],{i,ID}];
                    InsertVessel[ID,berth,arrival];
                    If ID == l Then
                         ReportSolution[tempCost];
                         λ = -1
                    Else
                         If tempCost + minVesselsCosts[] < minCost Then
                              Push[];
                              If tempCost + minVesselsCosts[] < minCost Then ID++
                                                                        Else Pop[]
                              EndIf
                         Else
                              λ = -1
                         EndIf
                    EndIf
               Else
                    λ = -1
               EndIf
          EndIf
EndWhile
```

# Configuration of the principal variables of the *Algorithm Opt 1*

| ID | 1 | 2 | 3 | 4 | 5 | | l-1 | l |
|---|---|---|---|---|---|---|---|---|
| | | | (8,11,115) | (7,10,96) | | | | |
| | | (1,9,100) | (8,11,115) | (6,10,96) | (8,11,115) | | | (8,11,115) |
| | (8,9,100) | (2,9,95) | (7,9,110) | (7,9,96) | (7,9,110) | . . . | (6,9,110) | (7,10,85) |
| | (8,9,100) | (1,8,85) | (7,9,110) | (6,9,96) | (7,9,110) | . . . | (6,8,110) | (7,10,80) |
| ξ | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | (3,1,30) | (6,3,15) | (5,4,20) | (5,1,8) | (7,7,6) | . . . | (6,4,10) | (5,4,10) |
| | (1,2,20) | (7,4,15) | (5,2,20) | (6,2,8) | (8,8,6) | . . . | (7,2,10) | (2,7,10) |
| | (2,1,20) | (5,4,15) | (4,3,20) | (5,3,8) | (6,8,6) | . . . | (5,3,10) | (1,8,10) |
| | (1,1,10) | (6,4,5) | (5,3,10) | (4,3,4) | (7,8,3) | . . . | (6,3,10) | (2,8,8) |
| *n* | *2* | *3* | *0* | *0* | *0* | . . . | | |
| | **20** | **15** | **10** | **4** | **3** | . . . | **10** | **8** |
| Σ | **TempCost = 35** | | ***minVesselsCost[]*** | | | | | |

# *Step Forward* – Selection of the position (5,3,10) for the ID=3 vessel

| | **ID** | **1** | **2** | **3** | **4** | **5** | | **l-1** | **l** |
|---|---|---|---|---|---|---|---|---|---|
| | | | | (8,11,115) | (7,10,96) | | | | |
| | | | (1,9,100) | (8,11,115) | (6,10,96) | (8,11,115) | | | (8,11,115) |
| | | (8,9,100) | (2,9,95) | (7,9,110) | (7,9,96) | (7,9,110) | . . . | (6,9,110) | (7,10,85) |
| | | (8,9,100) | (1,8,85) | (7,9,110) | (6,9,96) | (7,9,110) | . . . | (6,8,110) | (7,10,80) |
| $\xi$ | | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | | (3,1,30) | (6,3,15) | (5,4,20) | (5,1,8) | (7,7,6) | . . . | (6,4,10) | (5,4,10) |
| | | (1,2,20) | (7,4,15) | (5,2,20) | (6,2,8) | (8,8,6) | . . . | (7,2,10) | (2,7,10) |
| | | (2,1,20) | (5,4,15) | (4,3,20) | (5,3,8) | (6,8,6) | . . . | (5,3,10) | (1,8,10) |
| | | (1,1,10) | (6,4,5) | (5,3,10) | (4,3,4) | (7,8,3) | . . . | (6,3,10) | (2,8,8) |
| *n* | | *2* | *3* | *1* | *0* | *0* | . . . | | |
| | | **20** | **15** | **10** | **4** | **3** | . . . | **10** | **8** |
| $\Sigma$ | | **TempCost = 35** | | ***minVesselsCost[]*** | | | | | |

*Step Forward – Deletion* of the position (5,3,10) for vessels with ID>3

| ID | 1 | 2 | 3 | 4 | 5 | . . . | l-1 | l |
|---|---|---|---|---|---|---|---|---|
| | | | (8,11,115) | (7,10,96) | | | | |
| | | (1,9,100) | (8,11,115) | (6,10,96) | (8,11,115) | | | (8,11,115) |
| | (8,9,100) | (2,9,95) | (7,9,110) | (7,9,96) | (7,9,110) | . . . | (6,9,110) | (7,10,85) |
| | (8,9,100) | (1,8,85) | (7,9,110) | (6,9,96) | (7,9,110) | . . . | (6,8,110) | (7,10,80) |
| ξ | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | (3,1,30) | (6,3,15) | (5,4,20) | (5,1,8) | (7,7,6) | . . . | (6,4,10) | (5,3,10) |
| | (1,2,20) | (7,4,15) | (5,2,20) | (6,2,8) | (8,8,6) | . . . | (5,3,10) | (2,7,10) |
| | (2,1,20) | (5,4,15) | (4,3,20) | (5,3,8) | (6,8,6) | . . . | (7,2,10) | (1,8,10) |
| | (1,1,10) | (6,4,5) | (5,3,10) | (4,3,4) | (7,8,3) | . . . | (6,3,10) | (2,8,8) |
| *n* | *2* | *3* | *1* | *0* | *0* | . . . | | |
| | **20** | **15** | **10** | **4** | **3** | . . . | **10** | **8** |
| Σ | **TempCost = 35** | | ***minVesselsCost[]*** | | | | | |

*Step Forward* – Adjustment of the sums. Ready for the next step.

| ID | 1 | 2 | 3 | 4 | 5 | . . . | l-1 | l |
|---|---|---|---|---|---|---|---|---|
| | | | (8,11,115) | | | | | |
| | | (1,9,100) | (8,11,115) | (7,10,96) | (8,11,115) | | | |
| | (8,9,100) | (2,9,95) | (7,9,110) | (6,10,96) | (7,9,110) | . . . | | (8,11,115) |
| | (8,9,100) | (1,8,85) | (7,9,110) | (7,9,96) | (7,9,110) | . . . | (6,9,110) | (7,10,85) |
| ξ | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| | (3,1,30) | (6,3,15) | (5,4,20) | (4,3,8) | (7,7,6) | . . . | (8,4,10) | (3,7,15) |
| | (1,2,20) | (7,4,15) | (5,2,20) | (5,1,8) | (8,8,6) | . . . | (6,4,10) | (2,7,10) |
| | (2,1,20) | (5,4,15) | (4,3,20) | (6,2,8) | (6,8,6) | . . . | (7,2,10) | (1,8,10) |
| | (1,1,10) | (6,4,5) | (5,3,10) | (4,3,4) | (7,8,3) | . . . | (6,3,10) | (2,8,8) |
| *n* | *2* | *3* | *1* | *0* | *0* | . . . | | |
| | **20** | **15** | **10** | **4** | **3** | . . . | **10** | **8** |
| Σ | **TempCost = 45** | | | ***minVesselsCost[]*** | | | | |

# **T=15, m=8** – Brute Force Algorithm vs. Algorithm Opt 1

| *l* | Brute Force Algorithm | | Algorithm Opt 1 | | × |
|---|---|---|---|---|---|
| | Time | No. Sol. | Time | No. Sol. | |
| 6 | 1.341 | *3* | 0.094 | *2* | **14.266** |
| 7 | 7.550 | *11* | 0.546 | *8* | **13.828** |
| 8 | 13.712 | *25* | 0.702 | *7* | **19.533** |
| 9 | 19.687 | *25* | 1.076 | *10* | **18.296** |
| 10 | 128.981 | *30* | 12.823 | *6* | **10.058** |
| 11 | 357.340 | *224* | 45.458 | *10* | **7.861** |
| 12 | 472.728 | *54* | 59.982 | *11* | **7.881** |
| 13 | 651.644 | *345* | 100.448 | *18* | **6.487** |
| 14 | 847.050 | *550* | 130.556 | *16* | **6.488** |
| 15 | 2245.228 | *853* | 276.713 | *27* | **8.114** |

# *Algoritam Opt 2×2*

```
MinCost = +∞;
n  = Table[0, {ID,l}];
ID = 1;
λ  = 1;
stack = ξ;
While 0 < ID Do
          DeleteVessel[ID];
          If λ == -1 Then
               n[ID] = 0; ID--;
               Pop[];
               λ = 1,
          Else
               If n[ID] < Length[ξ[ID]] Then
                    n[ID]++;
                    berth     = ξ[ID,n[ID],1];
                    arrival   = ξ[ID,n[ID],2];
                    tempCost  = Sum[ξ[i,n[i],3],{i,ID}];
                    InsertVessel[ID,berth,arrival];
                    If ID == l Then
                         ReportSolution[tempCost];
                         λ = -1
                    Else
                         If tempCost + minVesselsCosts[] < minCost Then
                              Push[];
                              Heuristics[];
                              If tempCost + minVesselsCosts[] < minCost Then ID++
                                                                        Else Pop[]
                              EndIf
                         Else
                              λ = -1
                         EndIf
                    EndIf
               Else
                    λ = -1
               EndIf
          EndIf
EndWhile
```

# **T=15, m=8** – Algorithm Opt 1 vs. Algorithm Opt 2×2

| $l$ | Algorithm Opt 1 | | Algorithm Opt 2×2 | | × | $\times_{BF}$ |
|---|---|---|---|---|---|---|
| | Time | No. Sol. | Time | No. Sol. | | |
| 6 | 0.094 | *2* | 0.094 | *2* | **1.000** | **14.265** |
| 7 | 0.546 | *8* | 0.234 | *3* | **2.333** | **32.265** |
| 8 | 0.702 | *7* | 0.437 | *6* | **1.606** | **31.378** |
| 9 | 1.076 | *10* | 1.326 | *8* | **0.811** | **14.846** |
| 10 | 12.823 | *6* | 6.614 | *5* | **1.939** | **19.501** |
| 11 | 45.458 | *10* | 16.380 | *9* | **2.775** | **21.816** |
| 12 | 59.982 | *11* | 22.354 | *8* | **2.683** | **21.147** |
| 13 | 100.448 | *18* | 35.490 | *10* | **2.830** | **18.361** |
| 14 | 130.556 | *16* | 47.221 | *10* | **2.765** | **17.938** |
| 15 | 276.713 | *27* | 111.856 | *13* | **2.474** | **20.073** |

*CPLEX 11.2*

Linux Slackware 12, Kernel: 2.6.21.5
Intel Core 2 Duo CPU E6750 on 2.66GHz
RAM=8Gb

*Algorithm Opt 2×2*

Microsoft Windows 7 64bit OS
Wolfram Mathematica 8.0 (64 bit version)
HP Pavilion dv3 Computer
Intel Core i3 CPU M350 @2.27GHz
RAM=4Gb.

# **T=15, m=8** – CPLEX 11.2 vs. Algorithm Opt 2×2

| *l* | CPLEX 11.2 | Algorithm Opt 2×2 | | × |
|---|---|---|---|---|
| | Time | Time | No. Sol. | |
| 6 | 0.06 | 0.094 | *2* | **0.638** |
| 7 | 20.53 | 0.234 | *3* | **87.735** |
| 8 | 18.91 | 0.437 | *6* | **43.272** |
| 9 | 20.88 | 1.326 | *8* | **15.747** |
| 10 | 35.19 | 6.614 | *5* | **5.321** |
| 11 | | 16.380 | *9* | |
| 12 | 129.76 | 22.354 | *8* | **5.805** |
| 13 | 379.64 | 35.490 | *10* | **10.697** |
| 14 | | 47.221 | *10* | |
| 15 | 4588.20 | 111.856 | *13* | **41.019** |

# **T=54, m=12** – Algorithm Opt 1 vs. Algorithm Opt 2×2

| *l* | Algorithm Opt 1 | | Algorithm Opt 2×2 | | × |
|---|---|---|---|---|---|
| | Time | No. Sol. | Time | No. Sol. | |
| 21 | 9.360 | *12* | 3.416 | *2* | **2.740** |
| 22 | 29.640 | *12* | 3.837 | *2* | **7.725** |
| 23 | 55.785 | *17* | 4.852 | *4* | **11.497** |
| 24 | 58.157 | *17* | 5.054 | *4* | **11.507** |
| 25 | 210.522 | *20* | 5.694 | *6* | **36.972** |
| 26 | 3428.123 | *30* | 20.498 | *7* | **167.241** |
| 27 | | | 21.481 | *7* | |
| 28 | | | 88.795 | *9* | |
| 29 | | | 510.557 | *13* | |

## **T=54, m=12** – CPLEX 11.2 vs. Algorithm Opt 2×2

| *l* | CPLEX 11.2 | Algorithm Opt 2×2 | | × |
|---|---|---|---|---|
| | Time | Time | No. Sol. | |
| 21 | 14274<br>3:57:54 | 3.416 | *2* | **4178.57** |

In[698]:=

Out[698]=