Hard problem instances
of Bounded Diameter Minimum Spanning Tree, Generalised Minimum Spanning Tree, and Multidimensional Knapsack problems

Jano van Hemert
http://www.vanhemert.co.uk

Algorithms and Data Structures Group
Institute for Computer Graphics and Algorithms
Technical University of Vienna

Dissertanten Seminar 2005/12/15
Bounded Diameter Minimum Spanning Tree (BDMST)

**Definition**
- Given a full graph with weights set to Euclidean distance in a 2D space,
- find the spanning tree with minimum total weight,
- while respecting a maximum diameter constraint

**Representation**
- $n$ nodes,
- $(x, y)$ coordinates for each node
Evolutionary algorithm

Genetic operators

- **Uniform crossover**: with equal probability, choose each set of coordinates from either the first or second parent
- **Mutation**: with certain probability for each set of coordinates, generate uniform randomly a new set

Fitness

- Problems are solved using Martin Gruber’s 0-1 ILP formulation [1]
- The CPU-time to termination (= solution found and proven optimality) is maximised
Evolving BDMST problem instances
That are difficult to solve

First results

- Instances with $n = 20$ nodes
- Uniform random instances take on average a few seconds ($\approx 7$ seconds).
- After generating 11,600 problem instances, an instance that takes $\approx 352$ seconds was produced (123,942 nodes) (on an Intel Pentium 4 2.40GHz—4800 BogoMIPS)
- The solution is found after $\approx 35$ seconds (16,000 nodes)
- The same instance is also hard for another ILP formulation and takes very long
- Variable Neighbourhood Search [2] is not able to find the optimum
Evolving BDMST problem instances
That are difficult to solve

Analysis

- Observing 2-dimensional plots of points in series of improved instances
- Observing the consecutive series of optimal solutions for one instance
Evolving BDMST problem instances
That are difficult to solve

Analysis

- Observing 2-dimensional plots of points in series of improved instances
- Observing the consecutive series of optimal solutions for one instance

Hypothesis

Analysing several instances, we suspect the following characteristics to induce difficulty

1. nodes lie near the edges of the plane, and
2. nodes appear to be clustered into several groups
Specific BDMST instances

Perfect circle

- Layout 20 nodes at equal distance on a circle
- Result is a moderately hard instance (≈ 80 seconds)
Specific BDMST instances

<table>
<thead>
<tr>
<th>Perfect circle</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 nodes at equal distance on a circle</td>
</tr>
<tr>
<td>Result is a moderately hard instance ($\approx 80$ seconds)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Perturbed evolved instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alter the evolved instance by moving one arbitrary node to the centre</td>
</tr>
<tr>
<td>Result is a very easy instance ($\approx 4$ seconds)</td>
</tr>
</tbody>
</table>
### Specific BDMST instances

**Perfect circle**
- Layout 20 nodes at equal distance on a circle
- Result is a moderately hard instance ($\approx 80$ seconds)

**Perturbed evolved instance**
- Alter the evolved instance by moving one arbitrary node to the centre
- Result is a very easy instance ($\approx 4$ seconds)

**Intermediate solutions to the hard instance**
- Only a few updates of the solution
- A number of times the centre node of the tree is changed
Circular problem instance generator for BDMST

Procedure

- Lay $c$ clusters at uniform distances on a circle
- For each cluster, generate $n'$ nodes on the circle by adding a Gaussian random variable with a deviation of $\frac{2\pi}{c \cdot \text{tightness}}$
- Push each node away from the circle by adding another Gaussian random variable with deviation $\text{noise}$
Circular problem instance generator for BDMST

Procedure

- Lay \( c \) clusters at uniform distances on a circle
- For each cluster, generate \( n' \) nodes on the circle by adding a Gaussian random variable with a deviation of \( \frac{2\pi}{c \cdot \text{tightness}} \)
- Push each node away from the circle by adding another Gaussian random variable with deviation \( \text{noise} \)

Parameters

- **tightness** determines the average distance between nodes in a cluster, where higher is a smaller distance
- **noise** determines the amount of drift from the circle, where higher corresponds to further away
Circular problem instance generator for BDMST
Explained visually

Noise

Tightness
Circular problem instance generator for BDMST
Varying the tightness and noise values

circle-generator: 12 nodes, 3 clusters, 500 samples

Time (seconds)
Clustered problem instance generator for BDMST
Based on the idea of a cluster only

**Procedure**

- Choose $c$ cluster centres uniform randomly on a 2D plane
- For each cluster, generate $n'$ nodes by adding a Gaussian random variable with a deviation of $\frac{1}{c \cdot \text{tightness}}$

**Differences with circular version**

- Clusters can be anywhere
- Only one parameter to worry about
Clustered problem instance generator for BDMST
Varying the tightness for n=12, c=4
Clustered problem instance generator for BDMST
Varying the tightness for n=12, c=4

Statistical significance

- None of the results are normally distributed (KS-Lillefors-Test)
- Verification with Wilcoxon-Rank test (two-sided)
- Significance tests between $t_i$ and $t_{i+1}$
  - $t = 64$ and $t = 128$: $p(H0) = 0.000$
  - $t = 128$ and $t = 256$: $p(H0) = 0.278$
Clustered problem instance generator for BDMST

Varying the tightness for n=20,c=5

![Graph showing time (seconds) vs. tightness with 95% confidence intervals.]

- Cluster-generator: 20 nodes, 5 clusters, 105-170 samples
- 95% confidence intervals
Clustered problem instance generator for BDMST
Varying the tightness for n=20, c=5

Statistical significance

- None of the results are normally distributed (KS-Lillefors-Test)
- Verification with Wilcoxon-Rank test (two-sided)
- Significance tests between $t_i$ and $t_{i+1}$
- $t = 64$ and $t = 128$: $p(H0) = 0.0377$
- $t = 128$ and $t = 256$: $p(H0) = 0.109$
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST

Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST

Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST

Instances when varying the tightness (n=20,c=5)

Tightness = 32
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20, c=5)
Clustering problem instance generator for BDMST

Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20,c=5)

Tightness = 256

0  0.2  0.4  0.6  0.8  1

0  0.2  0.4  0.6  0.8  1

Tightness = 1

Tightness = 2

Tightness = 4

Tightness = 8

Tightness = 16

Tightness = 32

Tightness = 64

Tightness = 128

Tightness = 256

Tightness = 512

Tightness = 1024

Tightness = 2048
Clustered problem instance generator for BDMST

Instances when varying the tightness (n=20, c=5)
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20,c=5)
Clustered problem instance generator for BDMST
Instances when varying the tightness (n=20, c=5)
A new evolutionary algorithm for solving BDMST

<table>
<thead>
<tr>
<th>Main idea</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Novel representation based on the levels of nodes</td>
</tr>
<tr>
<td>- Decoding step to get a spanning tree uses code from Martin’s level optimisation VNS neighbourhood</td>
</tr>
<tr>
<td>- Special genetic operators that suit the representation</td>
</tr>
<tr>
<td>- Local improvement steps based on neighbourhoods of VNS [2]</td>
</tr>
</tbody>
</table>
A new evolutionary algorithm for BDMST
Genetic operators and decoding
A new evolutionary algorithm for BDMST
Local optimisation: edge exchange neighbourhood
A new evolutionary algorithm for BDMST

<table>
<thead>
<tr>
<th>Instance n</th>
<th>D nr.</th>
<th>time (sec)</th>
<th>$k_{\text{max}}$</th>
<th>VNS best</th>
<th>mean</th>
<th>stdev</th>
<th>level encoded EA best</th>
<th>mean</th>
<th>stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>20</td>
<td>1</td>
<td>50</td>
<td>25</td>
<td>17.753</td>
<td>18.108</td>
<td>0.12</td>
<td>16.953</td>
<td>17.048</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>17.688</td>
<td>17.966</td>
<td>0.10</td>
<td>16.826</td>
<td>16.940</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>17.799</td>
<td>18.114</td>
<td>0.10</td>
<td>16.947</td>
<td>17.072</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>17.930</td>
<td>18.161</td>
<td>0.11</td>
<td>17.044</td>
<td>17.140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>17.464</td>
<td>17.863</td>
<td>0.12</td>
<td>16.557</td>
<td>16.668</td>
</tr>
<tr>
<td>500</td>
<td>20</td>
<td>1</td>
<td>500</td>
<td>25</td>
<td>17.290</td>
<td>17.460</td>
<td>0.08</td>
<td>16.900</td>
<td>17.017</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>17.215</td>
<td>17.373</td>
<td>0.08</td>
<td>16.808</td>
<td>16.902</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>17.252</td>
<td>17.464</td>
<td>0.05</td>
<td>16.923</td>
<td>17.036</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>17.318</td>
<td>17.514</td>
<td>0.07</td>
<td>17.007</td>
<td>17.106</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>16.932</td>
<td>17.139</td>
<td>0.09</td>
<td>16.534</td>
<td>16.641</td>
</tr>
<tr>
<td>1000</td>
<td>25</td>
<td>1</td>
<td>100</td>
<td>50</td>
<td>25.850</td>
<td>26.188</td>
<td>0.13</td>
<td>25.238</td>
<td>25.564</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>25.501</td>
<td>25.981</td>
<td>0.17</td>
<td>25.104</td>
<td>25.318</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>25.340</td>
<td>25.705</td>
<td>0.09</td>
<td>24.839</td>
<td>25.097</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>25.562</td>
<td>26.128</td>
<td>0.17</td>
<td>25.346</td>
<td>25.592</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>25.504</td>
<td>25.826</td>
<td>0.15</td>
<td>24.956</td>
<td>25.236</td>
</tr>
<tr>
<td>1000</td>
<td>25</td>
<td>1</td>
<td>1000</td>
<td>50</td>
<td>25.177</td>
<td>25.572</td>
<td>0.14</td>
<td>23.788</td>
<td>23.963</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>25.015</td>
<td>25.342</td>
<td>0.14</td>
<td>23.635</td>
<td>23.793</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>24.816</td>
<td>25.086</td>
<td>0.11</td>
<td>23.434</td>
<td>23.574</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>25.289</td>
<td>25.572</td>
<td>0.11</td>
<td>23.837</td>
<td>23.984</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>25.026</td>
<td>25.254</td>
<td>0.12</td>
<td>23.464</td>
<td>23.669</td>
</tr>
</tbody>
</table>
Summary for BDMST

Evolving BDMST instances

- Successful in finding **small instances** that are **hard to solve**
- Common pattern in these instances used for creating two problem generators

Differences in performance between problem generators

- Problems for the circular version **are harder** than for the clusters only version
- Except when some points are really **near** to each other
- Clusters only version enforces a structure much **more generalisable**
Summary for BDMST

Novel evolutionary algorithm

- New representation with a decoder and specialised operators
- Uses one neighbourhood from VNS
- Very successful at solving benchmark instances
### Definition
- Given a full graph with weights (here, set to Euclidean distance in a 2D space),
- with the set of vertices partitioned into $c$ clusters,
- find the spanning tree with minimum total weight,
- where the tree contains exactly one vertex from each cluster

### Representation (same as for BDMST)
- $n$ nodes,
- $(x, y)$ coordinates for each node
Evolutionary algorithm
Similar to BDMST

**Genetic operators**

- **Uniform crossover**: with equal probability, choose each set of coordinates from either the first or second parent
- **Mutation**: with certain probability for each set of coordinates, generate uniform randomly a new set

**Fitness**

- Problems are solved using Bin Hu’s implementation of TODO: ....’s ILP formulation TODO: CITE!!
- The CPU-time to termination (= solution found and proven optimality) is maximised
Averaged improved series

Improved series: 30 points over 6 clusters, 20 samples

95% confidence intervals
Summary for GMST

Evolving GMST instances

- Some increase in difficulty is observed for small instances
- Analysing instances reveals unrealistic instances
# Multidimensional Knapsack Problem (MKP)

## Definition
- Given multiple sacks of different capacities, and items with a profit and a cost for each sack,
- maximise the profit without violating the capacities,
- by excluding and including items

## Representation
- $\vec{p}$: profits of the $n$ items
- $\vec{w}_j$ of length $n$: cost of items in each of $m$ sacks
- $\vec{r}$: capacity of the $m$ sacks
**Evolutionary algorithm**

### Genetic operators

- **Crossover**: none, as it highly disrupts the structure
- **Mutation**: with certain probability for each profit $p_i$, weight $w_{ji}$, and capacity $r_j$, add a variable drawn from a Gaussian distribution with a standard deviation of 5 (using floor)

### Fitness

- Problems are solved using ILP formulation, which employ a relaxed LP to find a bound [3]
- Maximise the time (in seconds) either to proof or to find the optimum
Evolving MKP instances
That are difficult to solve

First results

- Instances with 100 items and 5 constraints
- The CPU-time to termination of Cplex (= solution found and proven optimality) is maximised
- Large variation in uniform random instances, similar to the popular benchmarks
- After generating 11,600 problem instances, an instance that takes \( \approx 458 \) seconds was produced (845,233 nodes) (on an Intel Pentium 4 2.40GHz—4800 BogoMIPS)
Evolving MKP instances
That are difficult to solve

Analysis

• It is hard to search the space of problem instances, resulting in a small series of improved instances
• Instance takes $\approx 458$ seconds (845 233 nodes) to solve, but Cplex finds the solution much sooner at $\approx 3.40$ seconds (17 000 nodes)
• MKP instances are difficult to analyse,
  • no easy visualisation
  • few measures of hardness prediction, which have doubtful value anyway
Evolving MKP instances
That are difficult to solve

Ideas for improving performance

• Use more accurate performance measures, such as the number of nodes expanded by Cplex
• Use a completely different representation
• Create more diversity in initial population by generating random problem instances with a certain degree of correlation between costs and profits
Evolving MKP instances
That are difficult to solve

Ideas for improving interest level of instances

- Maximise the gap between the bound of the relaxed LP and the optimum (best so far $n=100,c=5$: $\approx 15$)
- Create instances that are difficult for one algorithm and easy for another (best difference so far 0.1%)
- Maximise the time or number of nodes required to find the optimum (best result so far for $n=100,c=5$: $\approx 52$ seconds or $210\,090$ nodes)
Evolving MKP instances
An improved series

![Graph](image)

**Evolved Multidimensional Knapsack problem instance**

- **Time to optimum**
- **Time to proof**

Time (seconds)
Improved instances

- **X** time to optimum
- **•** time to proof

The graph illustrates the evolution of MKP instances over time, showing the time to optimum and the time to proof for improved instances.
Evolving MKP instances
The distribution of profits during an improved series

![Graph showing the distribution of profits for different series of MKP instances.](image)
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series

---

**Diagram Description:**
- The diagram illustrates the distribution of profits over a series of improving instances.
- Each instance is labeled as `best-0` to `best-20`.
- The x-axis represents the profits, ranging from 70 to 1070.
- The y-axis indicates the frequency of occurrence for each profit level.
- The graph shows a concentration of profits around certain intervals, such as 70, 270, 470, 670, 870, and 1070, with `best-3` showing a notable peak near 870.
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances

The distribution of profits during an improved series

![Bar chart showing the distribution of profits for best-6 instances.](chart.png)
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series

best−0
Profits

best−1
Profits

best−2
Profits

best−3
Profits

best−4
Profits

best−5
Profits

best−6
Profits

best−7
Profits

best−8
Profits

best−9
Profits

best−10
Profits

best−11
Profits

best−12
Profits

best−13
Profits

best−14
Profits

best−15
Profits

best−16
Profits

best−17
Profits

best−18
Profits

best−19
Profits

best−20
Profits

Frequency

Profits

70 270 470 670 870 1070
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series

best-0
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-1
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-2
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-3
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-4
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-5
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-6
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-7
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-8
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-9
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12

best-10
Profits
70 270 470 670 870 1070
Frequency
0 2 4 6 8 10 12
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series

![Bar chart showing the distribution of profits for best-0 to best-15]
Evolving MKP instances
The distribution of profits during an improved series

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-0)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-1)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-2)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-3)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-4)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-5)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-6)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-7)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-8)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-9)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-10)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-11)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-12)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-13)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-14)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-15)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-16)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-17)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-18)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-19)

<table>
<thead>
<tr>
<th>Profits</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>270</td>
<td>2</td>
</tr>
<tr>
<td>470</td>
<td>4</td>
</tr>
<tr>
<td>670</td>
<td>6</td>
</tr>
<tr>
<td>870</td>
<td>8</td>
</tr>
<tr>
<td>1070</td>
<td>10</td>
</tr>
</tbody>
</table>

(best-20)
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series
Evolving MKP instances
The distribution of profits during an improved series

During an improved series

- The distribution of profits changes from uniform to a more Gaussian shape, with a peak at 70% of the maximum profit.
- The distribution converges into a stable form, although the underlying problems still change significantly.
- Only one of the righthand side values is changed when $9 \rightarrow 10$, which takes the time to optimum from $\approx 36$ to $\approx 40$ seconds.
- The constraints maintain a uniform distribution.
MKP instance generator
Instances with a known optimum

Main idea

- Create constraints and profits such that only one solution leads to an optimum
- Leave sufficient room for distraction to fool algorithms into trying wrong configurations
- These instances can be used to check how near algorithms can approximate the optimum
MKP instance generator
Forcing one solution

expensive    moderate    cheap
MKP instance generator
Forcing one solution

size limit
MKP instance generator

Procedure

- Create 3 groups of items: small $S$, medium $M$, and large $L$ of equal size $|S| = |M| = |L|
- Total items $n = |S| + |M| + |L|
- Profits: $p_S < p_M < p_L$
- Weights:
  - $w_S = (1, 2, \ldots, |S|)$
  - $w_M = ((n+1)/2, \ldots, (n+1)/2)$
  - $w_L = (n, n-1, \ldots, (n+1) - |L|)$
- Righthandsize:
  $$r_C = \sum_{i \in S} w_S(i) + \sum_{i \in L} w_L(i) = (n+1)|w_S|$$
- Optimal solution uses all items in $S$ and $L$; none from $M$
- Optimum is equal to $|S|p_S + |L|p_L = |S|(p_S + p_L)$
Example

- Create 3 groups of items: small, medium, and large of size 11
- Profits: $p_s = 20$, $p_l = 80$, and medium is $p_m = 33$
- Constraints:
  - $w_s = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$
  - $w_m = (17, 17, 17, 17, 17, 17, 17, 17, 17, 17)$
  - $w_l = (33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23)$
- Righthandsize: $r_C = 34 \cdot 11 = 374$
- Optimum solution yields profit of $(20 + 80) \cdot 11 = 1100$
### MKP instance generator

**Problem hardness**

<table>
<thead>
<tr>
<th>Complexity (nodes)</th>
<th>Size of the groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td>1000</td>
<td>30</td>
</tr>
<tr>
<td>10000</td>
<td>35</td>
</tr>
<tr>
<td>100000</td>
<td>40</td>
</tr>
<tr>
<td>1e+06</td>
<td>45</td>
</tr>
<tr>
<td>1e+07</td>
<td>50</td>
</tr>
<tr>
<td>1e+08</td>
<td>55</td>
</tr>
</tbody>
</table>

**Problem hardness with $p_s=20$, $p_m=33$, $p_m=80$**

- **time to proof**
- **time to optimum**

![Graph showing complexity vs size of the groups](image)
More is needed to pin down the optimum

- The following condition (I believe) does the trick,

\[ \forall a \in \{1, \ldots, |M|\}, I \subseteq S : a \cdot w_m \leq \sum_{i \in I} w_s(i) \Rightarrow a \cdot p_m < p_s \cdot |I| \]
More is needed to pin down the optimum

- The following condition (I believe) does the trick,
  \[ \forall a \in \{1, \ldots, |M|\}, I \subseteq S : a \cdot w_m \leq \sum_{i \in I} w_s(i) \Rightarrow a \cdot p_m < p_s \cdot |I| \]

- No need for checking every \( a \), because we know the best is
  \[ a = \left\lfloor \frac{\sum_{i \in I} w_s(i)}{w_m} \right\rfloor \]
MKP instance generator
Fixating the optimum

More is needed to pin down the optimum

- The following condition (I believe) does the trick,

\[ \forall a \in \{1, \ldots, |M|\}, I \subseteq S : a \cdot w_m \leq \sum_{i \in I} w_s(i) \Rightarrow a \cdot p_m < p_s \cdot |I| \]

- No need for checking every \(a\), because we know the best is

\[ a = \left\lfloor \frac{\sum_{i \in I} w_s(i)}{w_m} \right\rfloor \]

- However, we still need to check every \(I \subseteq S\)
MKP instance generator
Fixating the optimum

More is needed to pin down the optimum

- The following condition (I believe) does the trick,

\[
\forall a \in \{1, \ldots, |M|\}, I \subseteq S : a \cdot w_m \leq \sum_{i \in I} w_s(i) \Rightarrow a \cdot p_m < p_s \cdot |I|
\]

- No need for checking every \(a\), because we know the best is

\[
a = \left\lfloor \frac{\sum_{i \in I} w_s(i)}{w_m} \right\rfloor
\]

- However, we still need to check every \(I \subseteq S\)
- Idea: use an ILP formulation that maximises \(p_m\) without violating the above constraints
MKP instance generator
Interesting comparison

- D. Pisinger
- 3group generator

weights

profits

weights
## Summary for MKP

### Evolving problem instances

- A difficult process, with much variation in resulting instances
- Hard to find common properties, limited result from distribution of profits
Summary for MKP

Evolving problem instances

- A difficult process, with much variation in resulting instances
- Hard to find common properties, limited result from distribution of profits

Problem instance generator

- Known optimum, which is good for evaluation purposes
- Control of hardness not yet perfect, but some promising preliminary results
- Vary profits of items in order to get a more diverse efficiency ordering
Bounded Diameter Minimum Spanning Tree

- Tight clusters lead to long running times

Generalised Minimum Spanning Tree

- Instances are not realistic (sorry Bin!)

Multidimensional Knapsack

- Evolving difficult problem instances and analysing them is a difficult task
- Problem generator with known optimum is a step in the right direction
## Summary

### Bounded Diameter Minimum Spanning Tree
- Tight clusters lead to long running times

### Generalised Minimum Spanning Tree
- Instances are not realistic (sorry Bin!)
Summary

Bounded Diameter Minimum Spanning Tree

- Tight clusters lead to long running times

Generalised Minimum Spanning Tree

- Instances are not realistic (sorry Bin!)

Multidimensional Knapsack

- Evolving difficult problem instances and analysing them is a difficult task
- Problem generator with known optimum is a step in the right direction
Future research

Bounded Diameter Minimum Spanning Tree

- Experiments of new algorithms on instances from cluster problem generator
## Future research

<table>
<thead>
<tr>
<th>Bounded Diameter Minimum Spanning Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Experiments of new algorithms on instances from cluster problem generator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generalised Minimum Spanning Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Evolve instances that adhere to some real-world properties</td>
</tr>
</tbody>
</table>
## Future research

### Bounded Diameter Minimum Spanning Tree
- Experiments of new algorithms on instances from cluster problem generator

### Generalised Minimum Spanning Tree
- Evolve instances that adhere to some real-world properties

### Multidimensional Knapsack
- Finish proof for problem generator
- Evolve problem instances constrained by the problem generator template
- Test algorithms on large instances, how far are they from the optimum?
