A fast algorithm for a parametric assignment problem and applications to max-algebra

Elisabeth Gassner  Bettina Klinz

Graz University of Technology

Workshop on Combinatorial Optimization
Aussois, 2006
Contents

1. Introduction to max-algebra
2. The characteristic max-polynomial
3. A fast algorithm for a special parametric assignment problem
4. Generalizations and application to max-algebra
**Max-algebra**

**Definition**

Max-algebra is the algebraic system \((\mathbb{R} \cup \{-\infty\}, \otimes, \oplus)\) with

\[
a \otimes b := a + b \\
a \oplus b := \max(a, b)
\]
**Definition**

Max-algebra is the algebraic system \((\mathbb{R} \cup \{-\infty\}, \otimes, \oplus)\) with

\[
\begin{align*}
a \otimes b & := a + b \\
a \oplus b & := \max(a, b)
\end{align*}
\]

\[
A \otimes x = b
\]
Max-algebra is the algebraic system \((\mathbb{R} \cup \{-\infty\}, \otimes, \oplus)\) with

\[
a \otimes b := a + b
\]

\[
a \oplus b := \max(a, b)
\]
**Max-algebra**

**Definition**

Max-algebra is the algebraic system \((\mathbb{R} \cup \{-\infty\}, \otimes, \oplus)\) with

\[
a \otimes b := a + b \\
a \oplus b := \max(a, b)
\]

\[
A \otimes x = b \\
\max(a_{11} + x_1, a_{12} + x_2, \ldots, a_{1n} + x_n) = b_1 \\
\max(a_{21} + x_1, a_{22} + x_2, \ldots, a_{2n} + x_n) = b_2 \\
\vdots \\
\max(a_{n1} + x_1, a_{n2} + x_2, \ldots, a_{nn} + x_n) = b_n
\]

**Definition**

The characteristic max-polynomial $\chi_A(\lambda)$ of an $(n \times n)$-matrix $A$ is equal to the *maximal objective value of the linear assignment problem* with cost matrix

$$\begin{pmatrix}
\max(a_{11}, \lambda) & a_{12} & \cdots & a_{1n} \\
 a_{21} & \max(a_{22}, \lambda) & \cdots & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{n1} & a_{n2} & \cdots & \max(a_{nn}, \lambda)
\end{pmatrix}$$
Max-algebra

Definition

The characteristic max-polynomial $\chi_A(\lambda)$ of an $(n \times n)$-matrix $A$ is equal to the maximal objective value of the linear assignment problem with cost matrix

$$
\begin{pmatrix}
\max(a_{11}, \lambda) & a_{12} & \ldots & a_{1n} \\
a_{21} & \max(a_{22}, \lambda) & \ldots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \ldots & \max(a_{nn}, \lambda)
\end{pmatrix}
$$

piecewise linear cost coefficients!
**Max-algebra**

**Definition**

The characteristic max-polynomial \( \chi_A(\lambda) \) of an \( (n \times n) \)-matrix \( A \) is equal to the *maximal objective value of the linear assignment problem* with cost matrix

\[
\begin{pmatrix}
\max(a_{11}, \lambda) & a_{12} & \cdots & a_{1n} \\
 a_{21} & \max(a_{22}, \lambda) & \cdots & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{n1} & a_{n2} & \cdots & \max(a_{nn}, \lambda)
\end{pmatrix}
\]

piecewise linear cost coefficients!

R. E. Burkard, P. Butkovic: Max-algebra and the linear assignment problem (2003),
An Example

\[ c_\lambda(M) \]

\[ \lambda \]

E. Gassner (TU Graz)
An Example

![Diagram of a network with nodes and edges, showing an example of a parametric assignment problem. The diagram includes arrows and labels indicating the flow and costs.]

\[ c_\lambda(M) \]

\[ \lambda \]

E. Gassner (TU Graz)  Parametric Assignment Problem  January 2006 5 / 15
An Example

![Diagram showing a network with nodes and edges labeled with expressions involving λ. The graph includes a vertical axis labeled $c_\lambda(M)$ and a horizontal axis labeled $\lambda$. Nodes are connected with edges labeled by expressions like $7 - \lambda$, $5 - \lambda$, and $6$. The context suggests a discussion of a parametric assignment problem.](image-url)
An Example

**Diagram:**

- Nodes: 1, 2, 3, 4, 5, 6
- Edges and labels:
  - Node 1 to 4: $7 - \lambda$
  - Node 2 to 5: 6
  - Node 3 to 6: 1
  - Node 4 to 5: $5 - \lambda$

**Graph:**

- Directed edges connecting the nodes.

**Diagram:**

- $c_\lambda(M)$ vs. $\lambda$
- Two lines:
  - Blue line: descending
  - Red line: horizontal

**Equation:**

- $c_\lambda(M)$ function of $\lambda$
An Example

$\lambda$

$c_{\lambda}(M)$

$\lambda_1$
Problem Formulation

Given:

- balanced, bipartite graph $G = (U, V, E)$
- parametric edge weights $c_{\lambda}(i, j) = c(i, j) - \lambda b(i, j)$
**Problem Formulation**

**Given:**
- balanced, bipartite graph $G = (U, V, E)$
- parametric edge weights $c_\lambda(i, j) = c(i, j) - \lambda b(i, j)$

**Task:** Determine the optimal objective value function

$$z(\lambda) = \min \left\{ \sum_{(i,j)\in A} c_\lambda(i, j) \mid A \text{ is an assignment in } G \right\}$$

and the corresponding optimal assignments.
Problem Formulation

Given:
- balanced, bipartite graph $G = (U, V, E)$
- subset of edges $P \subseteq E$, parametric edge weights

$$c_{\lambda}(i, j) = \begin{cases} 
    c(i, j) - \lambda & \text{if } (i, j) \in P \text{ (parametric edge)} \\
    c(i, j) & \text{else (non-parametric edge)}
\end{cases}$$

Task: Determine the optimal objective value function

$$z(\lambda) = \min \left\{ \sum_{(i, j) \in A} c_{\lambda}(i, j) \mid A \text{ is an assignment in } G \right\}$$

and the corresponding optimal assignments.
Notation

assignment $M$ in $G$
parametric edges
Notation

parametric edges

incremental network $N(M)$
parametric edges

parametric backward edge
parametric edges

parametric forward edge
Notation

parametric edges

critical cycle $C = \text{zero-weight and augmenting}$
Theorem

Let $\lambda'$ be fixed and let $M'$ be an optimal assignment for $\lambda'$. Let $\delta \geq 0$ be minimal such that there exists a critical cycle $C$ in $N(M')$ at $\lambda' + \delta$. Then

- $M'$ is optimal for all $\lambda \in [\lambda', \lambda' + \delta]$,
- $M'' = M' \▽ C$ is also optimal at $\lambda' + \delta$ and contains more parametric edges than $M'$,
- $\lambda' + \delta$ is a breakpoint of $z(\lambda)$.
Algorithm (Version 1)

Step 1: Initialization
Algorithm (Version 1)

Step 1: Initialization

$\lambda_0 \leftarrow$ lower bound for first breakpoint
$M_0 \leftarrow$ optimal assignment for $\lambda_0$, $j \leftarrow 0$
Continue with Step 2.
Algorithm (Version 1)

Step 1: Initialization

\( \lambda_0 \leftarrow \) lower bound for first breakpoint
\( M_0 \leftarrow \) optimal assignment for \( \lambda_0 \), \( j \leftarrow 0 \)
Continue with Step 2.

Step 2: Pivot
Algorithm (Version 1)

Step 1: Initialization

\( \lambda_0 \leftarrow \text{lower bound for first breakpoint} \)

\( M_0 \leftarrow \text{optimal assignment for } \lambda_0, j \leftarrow 0 \)

Continue with Step 2.

Step 2: Pivot

if no critical cycle \( C \) exists in \( N(M_j) \) for \( \lambda \geq \lambda_j \) then

Stop.

else

Increase parameter until \( \exists \) critical cycle \( C \) in \( N(M_j) \) \( \Rightarrow \bar{\lambda} \)

Continue with Step 3.

end if
Algorithm (Version 1)

Step 1: Initialization
\[ \lambda_0 \leftarrow \text{lower bound for first breakpoint} \]
\[ M_0 \leftarrow \text{optimal assignment for } \lambda_0, j \leftarrow 0 \]
Continue with Step 2.

Step 2: Pivot
if no critical cycle \( C \) exists in \( N(M_j) \) for \( \lambda \geq \lambda_j \) then
   Stop.
else
   Increase parameter until \( \exists \) critical cycle \( C \) in \( N(M_j) \) \( \Rightarrow \bar{\lambda} \)
   Continue with Step 3.
end if

Step 3: New Assignment
Algorithm (Version 1)

**Step 1: Initialization**

\[ \lambda_0 \leftarrow \text{lower bound for first breakpoint} \]
\[ M_0 \leftarrow \text{optimal assignment for } \lambda_0, \ j \leftarrow 0 \]
Continue with Step 2.

**Step 2: Pivot**

if no critical cycle \( C \) exists in \( N(M_j) \) for \( \lambda \geq \lambda_j \) then
- Stop.
else
- Increase parameter until \( \exists \) critical cycle \( C \) in \( N(M_j) \) \( \Rightarrow \bar{\lambda} \)
- Continue with Step 3.
end if

**Step 3: New Assignment**

\[ M_{j+1} \leftarrow M_j \nabla C \text{ is new optimal assignment} \]
\[ \lambda_{j+1} \leftarrow \bar{\lambda} \text{ is a new breakpoint} \]
\[ j \leftarrow j + 1. \text{ Continue with Step 2.} \]
Algorithm (Version 1)

Step 1: Initialization
\( \lambda_0 \leftarrow \) lower bound for first breakpoint
\( M_0 \leftarrow \) optimal assignment for \( \lambda_0, j \leftarrow 0 \)
Continue with Step 2.

Step 2: Pivot

if no critical cycle \( C \) exists in \( N(M_j) \) for \( \lambda \geq \lambda_j \) then
Stop.
else
Increase parameter until \( \exists \) critical cycle \( C \) in \( N(M_j) \) \( \Rightarrow \bar{\lambda} \)
Continue with Step 3.
end if

Step 3: New Assignment

\( M_{j+1} \leftarrow M_j \triangledown C \) is new optimal assignment
\( \lambda_{j+1} \leftarrow \bar{\lambda} \) is a new breakpoint
\( j \leftarrow j + 1. \) Continue with Step 2.
Assignment $M$ is optimal

$\iff$

There is no negative-cost cycle $C$ in $N(M)$

$\iff$

There exists a shortest path tree in $N(M)$
**Theorem**

Assignment $M$ is optimal

$\iff$

There is no negative-cost cycle $C$ in $N(M)$

$\iff$

There exists a shortest path tree in $N(M)$

Idea:
Consider the parametric shortest path tree problem in $N(M)$. 
**Theorem**

Assignment $M$ is optimal

$\iff$

There is no negative-cost cycle $C$ in $N(M)$

$\iff$

There exists a shortest path tree in $N(M)$

Idea:
Consider the parametric shortest path tree problem in $N(M)$.

R. M. Karp and J. B. Orlin: Parametric shortest path algorithms with an application to cyclic staffing (1981),
N. E. Young, R. E. Tarjan, and J. B. Orlin: Faster parametric shortest path and minimum-balance algorithms (1991)
Assignments and shortest paths

Theorem

Assignment $M$ is optimal

$\iff$

There is no negative-cost cycle $C$ in $N(M)$

$\iff$

There exists a shortest path tree in $N(M)$

Idea:
Consider the parametric shortest path tree problem in $N(M)$.

R. M. Karp and J. B. Orlin: Parametric shortest path algorithms with an application to cyclic staffing (1981),
N. E. Young, R. E. Tarjan, and J. B. Orlin: Faster parametric shortest path and minimum-balance algorithms (1991)

$O(mn + n^2 \log n)$
Let $M'$ be optimal for $\lambda'$. 
Let $M'$ be optimal for $\lambda'$.

If $\exists$ shortest path tree for all $\lambda \geq \lambda'$ in $N(M')$ then

$\implies$ $M'$ is optimal for all higher parameter values. Stop.
Let $M'$ be optimal for $\lambda'$.

**if** $\exists$ shortest path tree for all $\lambda \geq \lambda'$ in $N(M')$ **then**

$\implies$ $M'$ is optimal for all higher parameter values. Stop.

**else**

Let $\bar{\lambda}$ be the maximal value such that there exists a shortest path tree in $N(M')$ for $\bar{\lambda}$.

$\implies$ $\exists$ critical cycle in $N(M')$ for $\bar{\lambda}$.

**end if**
Algorithm

Step 1: Initialization

\[ \lambda_0 \leftarrow \text{lower bound for first breakpoint} \]
\[ M_0 \leftarrow \text{optimal assignment for } \lambda_0, j \leftarrow 0 \]
Continue with Step 2.

Step 2: Pivot

if no critical cycle \( C \) exists in \( N(M_j) \) for \( \lambda \geq \lambda_j \) then
   Stop.
else
   Increase parameter until \( \exists \) critical cycle \( C \) in \( N(M_j) \) \( \Rightarrow \bar{\lambda} \)
   Continue with Step 3.
end if

Step 3: New Assignment

\[ M_{j+1} \leftarrow M_j \nabla C \text{ is new optimal assignment} \]
\[ \lambda_{j+1} \leftarrow \bar{\lambda} \text{ is a new breakpoint} \]
\[ j \leftarrow j + 1. \text{ Continue with Step 2.} \]
Step 2: Pivot

Build up the current incremental network $N(M_j)$.

$T \leftarrow$ shortest path tree in $N(M_j)$ for $\lambda_j$.

Solve parametric shortest path tree problem for $\lambda \geq \lambda_j$.

if $\exists$ shortest path tree for all $\lambda \geq \lambda_j$ in $N(M_j)$ then

Stop.

else

Let $\bar{\lambda}$ be the maximal value such that there exists a shortest path tree in $N(M_j)$ for $\bar{\lambda}$.

$\Rightarrow$ $\exists$ critical cycle in $N(M')$ for $\bar{\lambda}$.

Continue with Step 3.

end if
**Observation**

Let $T_j$ be a shortest path tree in $N(M_j)$ for $\lambda_j$. Then the tree $\tilde{T}_j$ in $N(M_{j+1})$ such that $T_j$ and $\tilde{T}_j$ have the same underlying undirected graph is a shortest path tree in $N(M_{j+1})$ for $\lambda_j$. 
Running Time Analysis

Observation

Let $T_j$ be a shortest path tree in $N(M_j)$ for $\lambda_j$. Then the tree $\tilde{T}_j$ in $N(M_{j+1})$ such that $T_j$ and $\tilde{T}_j$ have the same underlying undirected graph is a shortest path tree in $N(M_{j+1})$ for $\lambda_j$.

Observation

The number of path changes during the whole algorithm is bounded from above by $O(n^2)$. 

E. Gassner (TU Graz)
Observation

Let $T_j$ be a shortest path tree in $N(M_j)$ for $\lambda_j$. Then the tree $\tilde{T}_j$ in $N(M_{j+1})$ such that $T_j$ and $\tilde{T}_j$ have the same underlying undirected graph is a shortest path tree in $N(M_{j+1})$ for $\lambda_j$.

Observation

The number of path changes during the whole algorithm is bounded from above by $O(n^2)$.

Theorem

Our algorithm solves the parametric assignment problem in $O(mn + n^2 \log n)$ time.
**Generalizations**

**Given:**
- balanced, bipartite graph $G = (U, V, E)$
- piecewise affine linear, continuous edge weights with slopes in $\{-d, \ldots, d\} \subset \mathbb{Z}$ and $k$ breakpoints.
Given:
- balanced, bipartite graph \( G = (U, V, E) \)
- piecewise affine linear, continuous edge weights with slopes in \( \{-d, \ldots, d\} \subset \mathbb{Z} \) and \( k \) breakpoints.

Task:
- Find a minimal assignment for every value of the parameter.
Generalizations

Given:
- balanced, bipartite graph $G = (U, V, E)$
- piecewise affine linear, continuous edge weights with slopes in $\{-d, \ldots, d\} \subset \mathbb{Z}$ and $k$ breakpoints.

Task:
- Find a minimal assignment for every value of the parameter.
- Running time:
  $$\mathcal{O}(d(n + k)(m + n \log n))$$
Generalizations

Given:
- balanced, bipartite graph $G = (U, V, E)$
- piecewise affine linear, continuous edge weights with slopes in $\{-d, \ldots, d\} \subset \mathbb{Z}$ and $k$ breakpoints.

Task:
- Find a minimal assignment for every value of the parameter.
- Running time:
  
  $\mathcal{O}(d(n + k)(m + n \log n))$

Characteristic max-polynomial:
- piecewise affine linear, continuous edge weights with slopes in $\{-1, \ldots, 1\}$ and $n$ breakpoints (1 breakpoint for each diagonal element).
Generalizations

Given:
- balanced, bipartite graph \( G = (U, V, E) \)
- piecewise affine linear, continuous edge weights with slopes in \( \{-d, \ldots, d\} \subset \mathbb{Z} \) and \( k \) breakpoints.

Task:
- Find a minimal assignment for every value of the parameter.
- Running time:

\[ O(d(n + k)(m + n \log n)) \]

Characteristic max-polynomial:
- piecewise affine linear, continuous edge weights with slopes in \( \{-1, \ldots, 1\} \) and \( n \) breakpoints (1 breakpoint for each diagonal element).
- Running time:

\[ O(mn + n^2 \log n) \]
Thank you for your attention!