How to correctly prune tropical trees

Jean-Vincent Loddo
Luca Saiu

http://www-lipn.univ-paris13.fr/~loddo
http://www-lipn.univ-paris13.fr/~saiu

LIPN, Université Paris 13

Paris, July 5, 2010
Motivation

Many problems can be solved with a non-deterministic search on a space where some metric is optimized.
Motivation

Many problems can be solved with a non-deterministic search on a space where some metric is optimized.

- Tropical games are a generalization of min-max games
Motivation

Many problems can be solved with a non-deterministic search on a space where some metric is optimized.

- **Tropical games** are a generalization of min-max games
- Tropical games support **tropical $\alpha$-pruning**
  - tropical games whose dual is also tropical can be computed with $\alpha-\beta$ (not proved formally, but intuitive)
Many problems can be solved with a **non-deterministic search** on a space where some metric is **optimized**.

- **Tropical games** are a generalization of min-max games
- Tropical games support **tropical $\alpha$-pruning**
  - tropical games whose dual is also tropical can be computed with $\alpha$-$\beta$ (not proved formally, but intuitive)
- **Approximated parsing** is a tropical game
  - $\alpha$-pruning seems to be effective
  - ...we suspect many more problems to be tropical games
Combinatorial games (1): min-max

Let a game arena \( S = (\mathcal{P}, \lambda, \text{succ}) \) and a payoff function \( p : \mathcal{P}_T \to \mathbb{Z} \cup \{+\infty, -\infty\} \) be given.

The game value of a position \( \pi \in \mathcal{P} \) is traditionally defined as:

\[
v_p(\pi) = \begin{cases} 
p(\pi), & \pi \in \mathcal{P}_T \\ 
\min_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{P} \\ 
\max_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{O}
\end{cases}
\]
Combinatorial games (1): min-max

Let a game arena \( S = (P, \lambda, \text{succ}) \) and a payoff function \( p : P_T \to \mathbb{Z} \cup \{+\infty, -\infty\} \) be given.

The game value of a position \( \pi \in P \) is traditionally defined as:

\[
v_p(\pi) = \begin{cases} 
p(\pi), & \pi \in P_T \\
\min_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{P} \\
\max_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{O}
\end{cases}
\]

What if we generalize? let’s take a generic \( A = (\mathcal{U}, \oplus, \otimes) \) instead of \( (\mathbb{Z} \cup \{+\infty, -\infty\}, \min, \max) \)
Let a game arena $S = (\mathcal{P}, \lambda, \text{succ})$ and a payoff function $p : \mathcal{P}_T \rightarrow \mathcal{U}$ be given.

The game value of a position $\pi \in \mathcal{P}$ is defined as:

$$v_p(\pi) = \begin{cases} 
  p(\pi), & \pi \in \mathcal{P}_T \\
  \bigoplus_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{P} \\
  \bigodot_{i=1}^n v_p(\pi_i), & \text{succ}(\pi) = \langle \pi_1 \ldots \pi_n \rangle, \lambda(\pi) = \mathcal{O}
\end{cases}$$
Small-step semantics I

\[ \text{[Payoff]} \quad \frac{\pi \in \mathbb{P}_T}{\pi \rightarrow \mathcal{P}(\pi) = \mathcal{V}} \]

\[ \text{[\mathcal{P}\text{-expand}]} \quad \frac{suc\text{c}(\pi) = \vec{t}}{\pi \rightarrow \sum \vec{t}} \quad \lambda(\pi) = \mathcal{P} \quad \#\vec{t} \geq 1 \]

\[ \text{[\mathcal{O}\text{-expand}]} \quad \frac{suc\text{c}(\pi) = \vec{t}}{\pi \rightarrow \prod \vec{t}} \quad \lambda(\pi) = \mathcal{O} \quad \#\vec{t} \geq 1 \]

\[ \text{[\mathcal{P}\text{-reduce}]} \quad \sum \vec{t}\langle \mathcal{V}_1, \mathcal{V}_2 \rangle \vec{z} \rightarrow \sum \vec{t}\langle \mathcal{V} \rangle \vec{z} \quad \mathcal{V}_1 \oplus \mathcal{V}_2 = \mathcal{V} \]
Small-step semantics II

\[ [\mathcal{O}\text{-reduce}] \quad \prod \vec{t} \langle v_1, v_2 \rangle \vec{z} \rightarrow \prod \vec{t} \langle v \rangle \vec{z} \quad v_1 \odot v_2 = v \]

\[ [\text{Return}] \quad \Lambda \langle v \rangle \rightarrow v \]

\[ [\text{Context}] \quad \frac{t \rightarrow t'}{C[t] \rightarrow_c C[t']} \quad \text{for all contexts } C \]

The reflexive-transitive closure of $\rightarrow_c$ defines rewrites.
Tropical algebras

**Definition (Tropical algebra)**

\[ A = (\mathbb{U}, \oplus, \odot) \] is a *tropical algebra* if

- \( \oplus \) and \( \odot \) are associative
- \( \odot \) distributes over \( \oplus \): for all \( a, b, c \in \mathbb{U} \)
  
  \[ a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c) \] and
  
  \[ (a \oplus b) \odot c = (a \odot c) \oplus (b \odot c) \]

Slightly simplified: see the paper.
Tropical algebras

Definition (Tropical algebra)

\[ A = (\mathbb{U}, \oplus, \otimes) \] is a tropical algebra if

- \( \oplus \) and \( \otimes \) are associative
- \( \otimes \) distributes over \( \oplus \): for all \( a, b, c \in \mathbb{U} \)
  \[
  a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \quad \text{and} \quad
  (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)
  \]

Slightly simplified: see the paper.

The \( \text{min,} \, + \) algebra \((\mathbb{Z} \cup \{+\infty\}, \text{min,} \, +)\) is an important example of tropical algebra.
The *Rationality hypothesis*

For any $\alpha, \beta$ it’s easy to see that:

$$\alpha \leq \beta + \alpha$$
The *Rationality hypothesis*

For any $\alpha, \beta$ it’s easy to see that:

$$\alpha \leq \beta + \alpha$$

Or, using only the operations in $(\mathbb{Z} \cup \{+\infty\}, \text{min}, +)$:

$$\alpha = \text{min}\{\alpha, \beta + \alpha\}$$
The **Rationality hypothesis**

For any $\alpha, \beta$ it’s easy to see that:

$$\alpha \leq \beta + \alpha$$

Or, using only the operations in $(\mathbb{Z} \cup \{+\infty\}, \text{min}, +)$:

$$\alpha = \text{min}\{\alpha, \beta + \alpha\}$$

Generalizing to $(\mathbb{U}, \oplus, \odot)$ we obtain:

**Definition (Rationality)**

$A = (\mathbb{U}, \oplus, \odot)$ is *rational* iff for any $\alpha, \beta \in \mathbb{U}$ we have that

$$\alpha = \alpha \oplus (\beta \odot \alpha)$$

Slightly simplified: see the paper.
The *Rationality hypothesis*

For any $\alpha, \beta$ it’s easy to see that:

$$\alpha \leq \beta + \alpha$$

Or, using only the operations in $(\mathbb{Z} \cup \{+\infty\}, \min, +)$:

$$\alpha = \min\{\alpha, \beta + \alpha\}$$

Generalizing to $(\mathbb{U}, \oplus, \odot)$ we obtain:

**Definition (Rationality)**

$A = (\mathbb{U}, \oplus, \odot)$ is *rational* iff for any $\alpha, \beta \in \mathbb{U}$ we have that

$$\alpha = \alpha \oplus (\beta \odot \alpha)$$

Slightly simplified: see the paper.

“The player $P$ prefers $\alpha$ over $\alpha$ worsened by $\beta$”
The *Rationality hypothesis*

For any $\alpha, \beta$ it’s easy to see that:

$$\alpha \leq \beta + \alpha$$

Or, using only the operations in $(\mathbb{Z} \cup \{+\infty\}, \text{min}, +)$:

$$\alpha = \text{min}\{\alpha, \beta + \alpha\}$$

Generalizing to $(\mathbb{U}, \oplus, \odot)$ we obtain:

**Definition (Rationality)**

$\mathcal{A} = (\mathbb{U}, \oplus, \odot)$ is rational iff for any $\alpha, \beta \in \mathbb{U}$ we have that

$$\alpha = \alpha \oplus (\beta \odot \alpha)$$

“*The player $\mathcal{P}$ prefers $\alpha$ over $\alpha$ worsened by $\beta$”*

$\alpha$-pruning depends on *tropicality* and *rationality*
**Pruning trees**
Rewrite rules for pruning
Imperative-style algorithm

**P-cut**

Intuition: think of $\oplus$ as *min* and $\odot$ as $\cdot$. For any $v_0, v_1, t_0, t_1, \alpha, \beta$ if $\alpha \oplus \beta = \alpha$ then $t \leq t'$ ("$t'$ simulates $t$"):

```
\[
\begin{align*}
\alpha & \quad \beta \\
\quad & \odot v_0 \\
\quad & \oplus v_1 \\
\end{align*}
\]
```

"The player $P$ prefers $\alpha$ to $\beta$"
Intuition: think of $\oplus$ as $\text{min}$ and $\odot$ as $\text{+}$. For any $v_0$, $v_1$, $t_0$, $t_1$, $\alpha$, $\beta$ if $\alpha \oplus \beta = \alpha$ then $t \leq t'$ ("$t'$ simulates $t$"):

"The player $\mathcal{P}$ prefers $\alpha$ to $\beta$"
so computing $v_0$ is a waste of time
Intuition: think of $\oplus$ as $\min$ and $\odot$ as $\pm$. For any $v_0, v_1, t_0, t_1, \alpha, \beta$
if $\alpha \oplus \beta = \alpha$ then $t \leq t'$ ("$t'$ simulates $t$"):

"The player $\mathcal{P}$ prefers $\alpha$ to $\beta$"
so computing $v_0$ is a waste of time

if $\min\{\alpha, \beta\} = \alpha$ then $\min\{\alpha, (\beta + v_0), v_1\} = \min\{\alpha, v_1\}
Intuition: think of $\oplus$ as $\text{min}$ and $\odot$ as $\text{+}$. For any $v_0$, $v_1$, $t_0$, $t_1$, $\alpha$, $\beta$
if $\alpha \oplus \beta = \alpha$ then $t \leq t'$ ("$t'$ simulates $t$":)

"The player $\mathcal{P}$ prefers $\alpha$ to $\beta$"
so computing $v_0$ is a waste of time

if $\text{min}\{\alpha, \beta\} = \alpha$ then $\text{min}\{\alpha, (\beta + v_0), v_1\} = \text{min}\{\alpha, v_1\}$
if $\alpha \oplus \beta = \alpha$ then $\alpha \oplus (\beta \odot v_0) \oplus v_1 = \alpha \oplus v_1$
**P-will:** for any $t_0, t_1, t_2, \alpha, \beta$ we have $t \leq t'$

```
\begin{align*}
\alpha \oplus (\beta \odot v_0 \odot v_1) &\oplus v_2 \\
\min\{\alpha, (\beta + v_0 + v_1), v_2\} = \\
\min\{\alpha, (\beta + \min\{\alpha, v_0\} + v_1), v_2\} = \\
\alpha \oplus (\beta \odot (\alpha \oplus v_0) \odot v_1) \oplus v_2
\end{align*}
```

*This is due to rationality and properties of tropical algebras*

“The player $P$ can bequeath $\alpha$ to her grandchild”

(which allows to prune more)
The player $P$ can bequeath $\alpha$ to her grandchild”
(which allows to prune more)

$$\min\{\alpha, (\beta + v_0 + v_1), v_2\} = \min\{\alpha, (\beta + \min\{\alpha, v_0\} + v_1), v_2\}$$
\( \mathcal{P}\)-will: for any \( \vec{t}_0, \vec{t}_1, \vec{t}_2, \alpha, \beta \) we have \( t \leq t' \)

“The player \( \mathcal{P} \) can bequeath \( \alpha \) to her grandchild”

(which allows to prune more)

\[
\begin{align*}
\min\{\alpha, (\beta + v_0 + v_1), v_2\} &= \min\{\alpha, (\beta + \min\{\alpha, v_0\} + v_1), v_2\} \\
\alpha \odot (\beta \odot v_0 \odot v_1) \odot v_2 &= \alpha \odot (\beta \odot (\alpha \odot v_0) \odot v_1) \odot v_2
\end{align*}
\]
\[ P \text{-will: for any } \vec{t}_0, \vec{t}_1, \vec{t}_2, \alpha, \beta \text{ we have } t \leq t' \]

"The player \( P \) can bequeath \( \alpha \) to her grandchild"
(which allows to prune more)

\[
\min\{\alpha, (\beta + v_0 + v_1), v_2\} = \min\{\alpha, (\beta + \min\{\alpha, v_0\} + v_1), v_2\}
\]
\[
\alpha \oplus (\beta \odot v_0 \odot v_1) \oplus v_2 = \alpha \oplus (\beta \odot (\alpha \oplus v_0) \odot v_1) \oplus v_2
\]

{This is due to rationality and to properties of tropical algebras}
Pruning rules (for bi-tropical games)

- \[ \mathcal{P}\text{-will} \] \[ \sum \langle \alpha \left( \prod \langle \beta (\sum \vec{t}_1) \rangle \vec{t}_2 \right) \rangle \vec{t}_3 \rightarrow \sum \langle \alpha \left( \prod \langle \beta (\sum \langle \alpha \rangle \vec{t}_1) \rangle \vec{t}_2 \right) \rangle \vec{t}_3 \]

- \[ \mathcal{O}\text{-will} \] \[ \prod \langle \beta \left( \sum \langle \alpha (\prod \vec{t}_1) \rangle \vec{t}_2 \right) \rangle \vec{t}_3 \rightarrow \prod \langle \beta \left( \sum \langle \alpha \left( \prod \langle \beta \rangle \vec{t}_1 \right) \rangle \vec{t}_2 \right) \rangle \vec{t}_3 \]

- \[ \mathcal{P}\text{-cut} \] \[ \alpha \oplus \beta = \alpha \]

\[ \sum \langle \alpha \left( \prod \langle \beta \rangle \vec{t}_1 \right) \rangle \vec{t}_2 \rightarrow \sum \langle \alpha \rangle \vec{t}_2 \]

- \[ \mathcal{O}\text{-cut} \] \[ \beta \odot \alpha = \beta \]

\[ \prod \langle \beta \left( \sum \langle \alpha \rangle \vec{t}_1 \right) \rangle \vec{t}_2 \rightarrow \prod \langle \beta \rangle \vec{t}_2 \]
**α-β-pruning (1)** \([\alpha \text{ best till now for } \mathcal{P}/\min; \beta \text{ best till now for } \mathcal{O}/\max]\)

```plaintext
1  function alpha_beta(π : P; α, β : Z) : Z
2      if π ∈ \mathcal{P}_T then
3          return p(π)
4      π₁...πₙ := succ(π)  # n ≥ 1
5      if λ(π) = \mathcal{P} then
6          v := α  # Best for \mathcal{P} till now
7          for i from 1 to n
8              and while β < Z v do
9                  v := min{v, alpha_beta(πᵢ, v, β)}
10         else  # λ(π) = \mathcal{O}
11             v := β
12             for i from 1 to n
13                 and while v < Z α do
14                     v := max{v, alpha_beta(πᵢ, α, v)}
15      return v
```
Combinatorial games: from min-max to tropical

Tropical $\alpha$-pruning

Choose-How-To-Divide and Conquer

A tropical game example: parsing

$\alpha$-$\beta$-pruning (2) [$\alpha$ best till now for $P$/min; $\beta$ best till now for $O$/max]

```
function alpha_beta(\pi : P; \alpha, \beta : \mathbb{Z}) : \mathbb{Z}
    if \pi \in P_T then
        return p(\pi)
    \pi_1...\pi_n := succ(\pi) \# n \geq 1
    if \lambda(\pi) = P then
        v := \alpha \# Best for $P$ till now
        for i from 1 to n
            and while $\beta < \mathbb{Z} v$ do \# There's no $\beta$
                v := min\{v, alpha_beta(\pi_i, v, \beta)\} \# There's no $\beta$
    else \# $\lambda(\pi) = O$
        v := \beta \# There's no $\beta$
        for i from 1 to n
            and while $v < \mathbb{Z} \alpha$ do
                v := max\{v, alpha_beta(\pi_i, \alpha, v)\}
    return v
```
Tropical $\alpha$-pruning [\(\alpha\) current best for \(\mathcal{P}/\oplus\)]

1. function tropical(\(\pi: \mathcal{P}; \ \alpha: \mathbb{U}\)) : \mathbb{U}
2. \hspace{1em} if \(\pi \in \mathcal{P_T}\) then
3. \hspace{2em} return \(p(\pi)\)
4. \hspace{1em} \(\pi_1...\pi_n := \text{succ}(\pi) \ # n \geq 1\)
5. \hspace{1em} if \(\lambda(\pi) = \mathcal{P}\) then
6. \hspace{2em} \(v := \alpha\) \ # Best for \(\mathcal{P}\) till now
7. \hspace{2em} for \(i\) from 1 to \(n\) do
8. \hspace{3em} \# do not prune at \(\mathcal{P}\)'s level
9. \hspace{3em} \(v := v \oplus \text{tropical}(\pi_i, v)\)
10. \hspace{1em} else \ # \(\lambda(\pi) = \mathcal{O}\)
11. \hspace{2em} \(v := \text{tropical}(\pi_1, \alpha)\) \ # No \(1_U\)
12. \hspace{2em} for \(i\) from 2 to \(n\)
13. \hspace{3em} and while \(\alpha \oplus v \neq \alpha\) do
14. \hspace{4em} \(v := v \odot \text{tropical}(\pi_i, \alpha)\)
15. \hspace{1em} return \(v\)
Traditional D&C algorithms consist in:

- Divide a problem into sub-problems
- Combine solutions

For example, *merge-sort*. 
Divide and Conquer

Traditional D&C algorithms consist in:

- Divide a problem into sub-problems [deterministically]
- Combine solutions

For example, merge-sort.

Let’s generalize this to nondeterministic divisions.
Choose-How-To-Divide and Conquer

- non-deterministic choices in a solution space
- some way of “combining” sub-solutions.
non-deterministic choices in a solution space
choose a division \((\mathcal{P}, \oplus)\): for example \textit{min}

some way of “combining” sub-solutions.
Choose-How-To-Divide and Conquer

- non-deterministic choices in a solution space
  choose a division ($\mathcal{P}$, $\oplus$): for example $\min$
- some way of “combining” sub-solutions.
  “accumulating values” ($\mathcal{O}$, $\odot$): for example $+$
non-deterministic choices in a solution space
choose a division \((\mathcal{P}, \oplus)\): for example \(\text{min}\)

some way of “combining” sub-solutions.
“accumulating values” \((\mathcal{O}, \odot)\): for example +

If \(\mathcal{A} = (\mathbb{U}, \oplus, \odot)\) is tropical we can use \(\alpha\)-pruning
Choose-How-To-Divide and Conquer

- non-deterministic choices in a solution space
  - choose a division \((\mathcal{P}, \oplus)\): for example \textit{min}
  
- some way of “combining” sub-solutions.
  - “accumulating values” \((\mathcal{O}, \odot)\): for example \(+\)

If \(\mathcal{A} = (\mathcal{U}, \oplus, \odot)\) is tropical we can use \(\alpha\)-pruning

\(\oplus\) represents choice according to a quality criterion

\(\odot\) “accumulates”
 Parsing as a tropical game

Let a context-free grammar be given.

Find the “least-wrong” parse, where each parse may contain unmatched terminals
Parsing as a tropical game

Let a context-free grammar be given.

Find the “least-wrong” parse, where each parse may contain unmatched terminals

- Minimum number of errors
- Minimum number of unmatched characters
Parsing as a tropical game

Let a context-free grammar be given.

Find the “least-wrong” parse, where each parse may contain unmatched terminals

- Minimum number of errors
- Minimum number of unmatched characters

Hypotheses to make exposition simpler (liftable):

- No $\epsilon$-productions
- No two adjacent non-terminals
- No non-terminal alone on a right side
Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example

"1 + 2 + 3" with $E$. 

Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example

"1 + 2 + 3" with $E$.

- $\mathcal{P}$: Choose a production and a division (pivot characters)
Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example

"1 + 2 + 3" with $E$.

- $P$: Choose a production and a division (pivot characters)
- $O$: Parse nonterminals “in between” pivots
Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example

"1 + 2 + 3" with $E$.

- $\mathcal{P}$: Choose a production and a division (pivot characters)
- $\mathcal{O}$: Parse nonterminals “in between” pivots

$\pi \mathcal{P} = ("1 + 2 + 3", E)$
Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example

"1 + 2 + 3" with $E$.

- $\mathcal{P}$: Choose a production and a division (pivot characters)
- $\mathcal{O}$: Parse nonterminals “in between” pivots

$$\pi_\mathcal{P} = ("1 + 2 + 3", E)$$
$$\pi_\mathcal{O} = ("1", E), ("2 + 3", E)$$
Parsing as a tropical game: nondeterministic parsing

We have a substring to parse with a nonterminal, for example "1 + 2 + 3" with $E$.

- $P$: Choose a production and a division (pivot characters)
- $O$: Parse nonterminals "in between" pivots

\[
\begin{align*}
\pi_P &= ("1 + 2 + 3", E) \\
\pi_O &= ("1", E), ("2 + 3", E)
\end{align*}
\]

$A = (\mathbb{Z} \cup \{+\infty\}, \text{min}, +)$
Parsing as a tropical game: example

Parsing "1 + 2 + 3":

\[
E ::= n \\
E ::= v \\
E ::= ( E ) \\
E ::= \text{let } v = E \text{ in } E \\
E ::= \text{if } E \text{ then } E \text{ else } E \\
E ::= E = E \\
E ::= E + E \\
E ::= E \ast E
\]
Parsing "1 + 2 + 3":

\[ E ::= n \]
\[ E ::= v \]
\[ E ::= (E) \]
\[ E ::= \text{let } v = E \text{ in } E \]
\[ E ::= \text{if } E \text{ then } E \text{ else } E \]
\[ E ::= E = E \]
\[ E ::= E + E \]
\[ E ::= E \times E \]

What if we had \texttt{let} instead of \texttt{3}? An error...
Memoization

Memoization: avoiding to re-examined repeated positions
Preliminary practical test

Is $\alpha$-pruning useful?

Is memoization useful?

[little demo]
Other $\alpha$-pruning examples

- Searching the most general solution in logic programming
  [Loddo-DiCosmo, LPAR 2000] [Loddo PhD 2002]
Other $\alpha$-pruning examples

- Searching the most general solution in logic programming [Loddo-DiCosmo, LPAR 2000] [Loddo PhD 2002]
  - optimization criterion: “less-instantiated terms”. $inf$, $sup$ bi-tropical
Other $\alpha$-pruning examples

- Searching the most general solution in logic programming [Loddo-DiCosmo, LPAR 2000] [Loddo PhD 2002]
  - optimization criterion: "less-instantiated terms". $\inf$, $\sup$ bi-tropical

- Draw a planar graph...
Other $\alpha$-pruning examples

- Searching the most general solution in logic programming [Loddo-DiCosmo, LPAR 2000] [Loddo PhD 2002]
  - optimization criterion: “less-instantiated terms”. $\inf$, $\sup$
  - bi-tropical

- Draw a planar graph...

...minimizing:
  - the occupied area
Other $\alpha$-pruning examples

- Searching the most general solution in logic programming [Loddo-DiCosmo, LPAR 2000] [Loddo PhD 2002]
  - optimization criterion: “less-instantiated terms”. $\inf$, $\sup$
  - bi-tropical

- Draw a planar graph...
  
  ![Planar Graph](image)

...minimizing:

- the occupied area
- the number of crossings
Conclusions

We...

- defined and proved correct tropical $\alpha$-pruning
- formalized min-max games as particular (bi-)tropical games
- modeled approximated parsing as a tropical game
  - $\alpha$-pruning seems to be effective

We would like...

- to find more applications
- more realistic implementations
Conclusions

We...

- ...defined and proved correct "tropical $\alpha$-pruning"
- ...formalized "min-max games as particular (bi-)tropical games"
- ...modeled "approximated parsing as a tropical game"
  - $\alpha$-pruning seems to be effective

We would like...

- ...to find more applications
- ...more realistic implementations

Thanks!