Efficient Computation of Throughput Values of Context-Free Languages

Didier Caucal[1] Jurek Czyzowicz[2] Wojciech Fraczak[2]
Wojciech Rytter[3]

IGM-CNRS, Marne-la-Vallée, France

Dépt d'informatique, Université du Québec en Outaouais, Gatineau PQ, Canada

Inst. of Informatics, Warsaw University, Warsaw, Poland

CIAA 2007, Prague, Czech Republic

Introduction and Motivation

In the context of system-performance analysis:

Throughput

=

the measure of the worst case speed of processing data

=

the lower bound (the infimum) of the greatest ratio of the length of processed input to its processing time, taken over all possible computations

very important in the context of network packet processing

Introduction and Motivation

In the context of system-performance analysis:

Throughput

=

the measure of the worst case speed of processing data

=

the lower bound (the infimum) of the greatest ratio of the length of processed input to its processing time, taken over all possible computations

very important in the context of network packet processing

Introduction and Motivation

In the context of system-performance analysis:

Throughput

=

the measure of the worst case speed of processing data

=

the lower bound (the infimum) of the greatest ratio of the length of processed input to its processing time, taken over all possible computations

very important in the context of network packet processing

Introduction and Motivation

In the context of system-performance analysis:

Throughput

=

the measure of the worst case speed of processing data

=

the lower bound (the infimum) of the greatest ratio of the length of processed input to its processing time, taken over all possible computations

very important in the context of network packet processing

Introduction and Motivation

In the context of system-performance analysis:

Throughput

=

the measure of the worst case speed of processing data

=

the lower bound (the infimum) of the greatest ratio of the length of processed input to its processing time, taken over all possible computations

very important in the context of network packet processing

- System seen as a set L representing all its execution traces.
- One trace, $w \in L$, can be:

$$\bullet \xrightarrow[3bits/1sec]{task\ 1} \bullet \xrightarrow[0bits/2sec]{task\ 2} \bullet \xrightarrow[10bits/1sec]{task\ 3} \bullet \xrightarrow[2bits/1sec]{task\ 4} \bullet$$

  The throughput of a trace is its *mean weight*, i.e.:

$$\overline{\rho}(w) \stackrel{\text{def}}{=} \frac{\rho(w)}{|w|} = \frac{3+0+10+2}{1+2+1+1} = 3bits/sec$$

- The throughput of the system is defined as the infimum of the mean weight of all traces of L:

$$throughput(L) \stackrel{\text{def}}{=} \inf \{\overline{\rho}(w) \mid w \in L\}.$$

- System seen as a set L representing all its execution traces.
- One trace, $w \in L$, can be:

$$\bullet \xrightarrow[3bits/1sec]{task\ 1} \bullet \xrightarrow[0bits/2sec]{task\ 2} \bullet \xrightarrow[10bits/1sec]{task\ 3} \bullet \xrightarrow[2bits/1sec]{task\ 4} \bullet$$

The throughput of a trace is its *mean weight*, i.e.:

$$\overline{\rho}(w) \stackrel{\text{def}}{=} \frac{\rho(w)}{|w|} = \frac{3+0+10+2}{1+2+1+1} = 3bits/sec$$

- The throughput of the system is defined as the infimum of the mean weight of all traces of L:

$$throughput(L) \stackrel{\text{def}}{=} \inf \{\overline{\rho}(w) \mid w \in L\}.$$

- System seen as a set L representing all its execution traces.
- One trace, $w \in L$, can be:

$$\bullet \xrightarrow[3bits/1sec]{\textit{task 1}} \bullet \xrightarrow[0bits/2sec]{\textit{task 2}} \bullet \xrightarrow[10bits/1sec]{\textit{task 3}} \bullet \xrightarrow[2bits/1sec]{\textit{task 4}} \bullet$$

The throughput of a trace is its *mean weight*, i.e.:

$$\overline{\rho}(w) \stackrel{\text{def}}{=} \frac{\rho(w)}{|w|} = \frac{3+0+10+2}{1+2+1+1} = 3bits/sec$$

- The throughput of the system is defined as the infimum of the mean weight of all traces of L:

$$throughput(L) \stackrel{\text{def}}{=} \inf \{\overline{\rho}(w) \mid w \in L\}.$$

- System seen as a set L representing all its execution traces.
- One trace, $w \in L$, can be:

$$\bullet \xrightarrow[3bits/1sec]{task\ 1} \bullet \xrightarrow[0bits/2sec]{task\ 2} \bullet \xrightarrow[10bits/1sec]{task\ 3} \bullet \xrightarrow[2bits/1sec]{task\ 4} \bullet$$

The throughput of a trace is its *mean weight*, i.e.:

$$\overline{\rho}(w) \stackrel{\text{def}}{=} \frac{\rho(w)}{|w|} = \frac{3+0+10+2}{1+2+1+1} = 3bits/sec$$

- The throughput of the system is defined as the infimum of the mean weight of all traces of L:

$$throughput(L) \stackrel{\text{def}}{=} \inf \{\overline{\rho}(w) \mid w \in L\}.$$

- A simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of data.

  In such a case any standard algorithm for minimum mean cycle calculation can be used

- When more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems.

  The throughput of a system is better approximated by the context-free grammar throughput computation.

- In this talk we are concerned with an Efficient Computation of Throughput of Context-Free Languages.

- A simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of data.

  In such a case any standard algorithm for minimum mean cycle calculation can be used

- When more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems.

  The throughput of a system is better approximated by the context-free grammar throughput computation.

- In this talk we are concerned with an Efficient Computation of Throughput of Context-Free Languages.

- A simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of data.

  In such a case any standard algorithm for minimum mean cycle calculation can be used

- When more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems.

  The throughput of a system is better approximated by the context-free grammar throughput computation.

- In this talk we are concerned with an Efficient Computation of Throughput of Context-Free Languages.

- A simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of data.

  In such a case any standard algorithm for minimum mean cycle calculation can be used

- When more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems.

  The throughput of a system is better approximated by the context-free grammar throughput computation.

- In this talk we are concerned with an Efficient Computation of Throughput of Context-Free Languages.

- A simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of data.

  In such a case any standard algorithm for minimum mean cycle calculation can be used

- When more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems.

  The throughput of a system is better approximated by the context-free grammar throughput computation.

- In this talk we are concerned with an Efficient Computation of Throughput of Context-Free Languages.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $throughput(L) = throughput(R)$.

⊖ Expensive!

A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $throughput(L) = throughput(R)$.

⊖ Expensive!

A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $throughput(L) = throughput(R)$.

⊖ Expensive!

A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $throughput(L) = throughput(R)$.
- Expensive!

  A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $\mathit{throughput}(L) = \mathit{throughput}(R)$.

⊖ Expensive!

A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

Via Parikh theorem:

- Parikh showed that the commutative image of every context-free language is the commutative image of some regular language.
- The mean weight of a word w is independent of the order of symbols used in w. Hence, the alphabet commutativity may be used.
- Consequently, for every context-free language L we can find a regular language R such that $throughput(L) = throughput(R)$.

⊖ Expensive!

A direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G.

Existing solutions

By approximation:

- In 2005, we gave an $O(n^2 \log(\frac{\max - \min}{\epsilon}))$ algorithm, computing an ϵ-approximation of the throughput of a CFL, where n is the size of the grammar, and max (min) is the maximum (respectively, minimum) letter weight of the alphabet.

⊕ A satisfactory solution from a practical point of view

Existing solutions

By approximation:

- In 2005, we gave an $O(n^2 \log(\frac{\max - \min}{\epsilon}))$ algorithm, computing an ϵ-approximation of the throughput of a CFL, where n is the size of the grammar, and max (min) is the maximum (respectively, minimum) letter weight of the alphabet.

⊕ A satisfactory solution from a practical point of view

Existing solutions

By approximation:

- In 2005, we gave an $O(n^2 \log(\frac{\max - \min}{\epsilon}))$ algorithm, computing an ϵ-approximation of the throughput of a CFL, where n is the size of the grammar, and max (min) is the maximum (respectively, minimum) letter weight of the alphabet.

$\oplus$ A satisfactory solution from a practical point of view

Outline of our algorithm

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G.

1. Compute G_2, a 2-reduced form for G.
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$, i.e., $throughput(L(G')) = throughput(L(G))$.
3. Find the throughput of the finite language $L(G')$.

Outline of our algorithm

Algorithm "Throughput-Calculation"
INPUT: Context-free grammar G.

1. Compute G_2, a 2-reduced form for G.
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$, i.e., $throughput(L(G')) = throughput(L(G))$.
3. Find the throughput of the finite language $L(G')$.

Outline of our algorithm

Algorithm "Throughput-Calculation"
INPUT: Context-free grammar G.

1. Compute G_2, a 2-reduced form for G.
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$, i.e., $throughput(L(G')) = throughput(L(G))$.
3. Find the throughput of the finite language $L(G')$.

Outline of our algorithm

Algorithm “Throughput-Calculation”
INPUT: Context-free grammar G.

1. Compute G_2, a 2-reduced form for G.
2. Compute a throughput invariant finite language grammar $G' = \mathit{Fin}(G_2)$, i.e., $\mathit{throughput}(L(G')) = \mathit{throughput}(L(G))$.
3. Find the throughput of the finite language $L(G')$.

Notation and assumptions

- $G = (\Sigma, N, P, S)$ — a context-free grammar

  Context-free grammars under consideration are 2-reduced grammars. I.e.:
  - trimmed (there are no useless nonterminals); and
  - each of its production rules has one or two symbols on the right-hand side.
- ρ — weight function $\rho : \Sigma \to \mathbb{N}$

Notation and assumptions

- $G = (\Sigma, N, P, S)$ — a context-free grammar

  Context-free grammars under consideration are 2-reduced grammars. I.e.:
  - trimmed (there are no useless nonterminals); and
  - each of its production rules has one or two symbols on the right-hand side.
- ρ — weight function $\rho : \Sigma \to \mathbb{N}$

Notation and assumptions

- $G = (\Sigma, N, P, S)$ — a context-free grammar

  Context-free grammars under consideration are 2-reduced grammars. I.e.:
  - trimmed (there are no useless nonterminals); and
  - each of its production rules has one or two symbols on the right-hand side.
- ρ — weight function $\rho : \Sigma \to \mathbb{N}$

Notation and assumptions

- $G = (\Sigma, N, P, S)$ — a context-free grammar

  Context-free grammars under consideration are 2-reduced grammars. I.e.:

  - trimmed (there are no useless nonterminals); and
  - each of its production rules has one or two symbols on the right-hand side.

- ρ — weight function $\rho : \Sigma \to \mathbb{N}$

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $throughput(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \longmapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $\mathit{throughput}(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $\mathit{throughput}(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $throughput(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $throughput(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $\mathit{throughput}(L) \geq t$.

- Given a positive real value t, we define *throughput balance* of L with respect to t as: $tb(L,t) \stackrel{\text{def}}{=} \min\{(\rho(w) - |w|t) \mid w \in L\}$.
- If $tb(L,t) > 0$ (resp., $tb(L,t) < 0$) then language L has a "surplus" (resp., "deficit") in achieving througput t.
- $tb(L,t)$ corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $\mathit{throughput}(L) \geq t$.
- Let L be a finite language and $m \in \mathbb{N}$ the maximum length of a word of L. The minimum difference between mean weight of two words of L is not smaller than $\frac{1}{m^2}$.

Theorem. There exists an $O(n \log md_\rho)$ time algorithm that computes $\mathit{throughput}(L)$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $\mathit{throughput}(L) \geq t$.
- Let L be a finite language and $m \in \mathbb{N}$ the maximum length of a word of L. The minimum difference between mean weight of two words of L is not smaller than $\frac{1}{m^2}$.

Theorem. There exists an $O(n \log md_\rho)$ time algorithm that computes $\mathit{throughput}(L)$.

Throughput of a finite language

Observations:

- In the case of a finite language L there always exists a word in L, whose mean weight equals the throughput of L.
- Given a grammar G of size n for a finite language L and a positive real value t, we can decide in $O(n)$ time whether $throughput(L) \geq t$.
- Let L be a finite language and $m \in \mathbb{N}$ the maximum length of a word of L. The minimum difference between mean weight of two words of L is not smaller than $\frac{1}{m^2}$.

Theorem. There exists an $O(n \log md_\rho)$ time algorithm that computes $throughput(L)$.

Throughput invariant grammar transformation

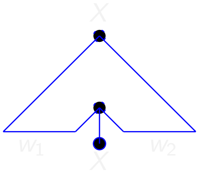
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \to w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

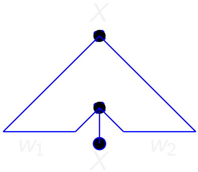
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \rightarrow w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

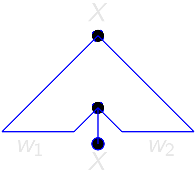
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \rightarrow w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

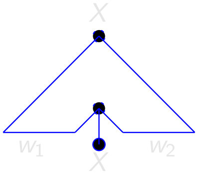
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \to w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

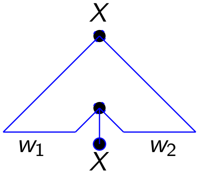
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \rightarrow w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

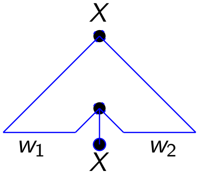
Main observation:

The throughput of $L(G)$ is either

1. equal to the mean weight of some word $w \in L(G)$, whose syntax tree is not recursive (i.e., at most of depth $|N|$);

   The set $L_f(G)$ of such words is finite

2. or equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, such that there exists a derivation $X \to w_1 X w_2$ with the corresponding syntax tree of depth not bigger than $|N|$.



The set $L_r(G)$ of such words is finite

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput invariant grammar transformation

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n.

- There exists a grammar G_f of size $O(n^2)$ defining $L_f(G)$.
- There exists a grammar G_r of size $O(n^3)$ defining $L_r(G)$.

Let $G' = Fin(G)$ be the disjoint union of G_f and G_r.

- G' defines a finite language and is of size $O(n^3)$
- $throughput(L(G')) = throughput(L(G))$
- All syntax trees of G' are of heigh at most $|N| + 1$, i.e., the longest word generated by G' is $2^{|N|}$.

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.

   — in $O(n)$ time, yielding G_2 of size $O(n)$ —

2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$

   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —

3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.

   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Throughput computation of a context-free language

Algorithm "Throughput-Calculation"

INPUT: Context-free grammar G of size n.

1. Compute G_2, a 2-reduced form for G.
   — in $O(n)$ time, yielding G_2 of size $O(n)$ —
2. Compute a throughput invariant finite language grammar $G' = Fin(G_2)$
   — in $O(n^3)$ time, yielding G' of size $O(n^3)$ —
3. Find the throughput of $L(G')$ and report it as the throughput of $L(G)$.
   — in $O(n^4 + n^3 \log d)$ time —

Conclusions

- We presented the first polynomial-time algorithm computing the throughput of context-free languages

  The problem may be seen as a generalization of the minimal mean weight cycle problem for finite digraphs, to the case of the class of graphs generated by context-free grammars.

- Unfortunately the complexity of our approach is still high from the practical point of view.

  How to improve the proposed solution, possibly by using a completely different approach exploiting explicitly the commutative property?

Conclusions

- We presented the first polynomial-time algorithm computing the throughput of context-free languages

  The problem may be seen as a generalization of the minimal mean weight cycle problem for finite digraphs, to the case of the class of graphs generated by context-free grammars.

- Unfortunately the complexity of our approach is still high from the practical point of view.

  How to improve the proposed solution, possibly by using a completely different approach exploiting explicitly the commutative property?

Conclusions

- We presented the first polynomial-time algorithm computing the throughput of context-free languages

  The problem may be seen as a generalization of the minimal mean weight cycle problem for finite digraphs, to the case of the class of graphs generated by context-free grammars.

- Unfortunately the complexity of our approach is still high from the practical point of view.

  How to improve the proposed solution, possibly by using a completely different approach exploiting explicitly the commutative property?

Conclusions

- We presented the first polynomial-time algorithm computing the throughput of context-free languages

  The problem may be seen as a generalization of the minimal mean weight cycle problem for finite digraphs, to the case of the class of graphs generated by context-free grammars.

- Unfortunately the complexity of our approach is still high from the practical point of view.

  How to improve the proposed solution, possibly by using a completely different approach exploiting explicitly the commutative property?