Double digest (DDP) a simplified partial digest problem (SPDP) benchmarky

Michal Mašek
Martina Tomisová

Co vás čeká

- Měření času v Pythonu

Double digest

- Připomenutí algoritmu
- Metodika měření
- Výsledky

Simplified partial digest

- Připomenutí algoritmu
- Metodika měření
- Výsledky

Měření času v Pythonu

- Funce `Timeit` z modulu `timeit`
- Pomocí odečítání výsledků funkce `time` z modulu `time`

Timeit

```
from timeit import Timer

# testovaná funkce

t = Timer("double_digest_test()", "from __main__
import double_digest_test")

# funkce se zavolá reps_cnt krát na stejná data

time_for_all = t.timeit(reps_cnt)

# jak dlouho trvalo jedno proběnutí funkce

time_for_one = time_for_all/reps_cnt
```

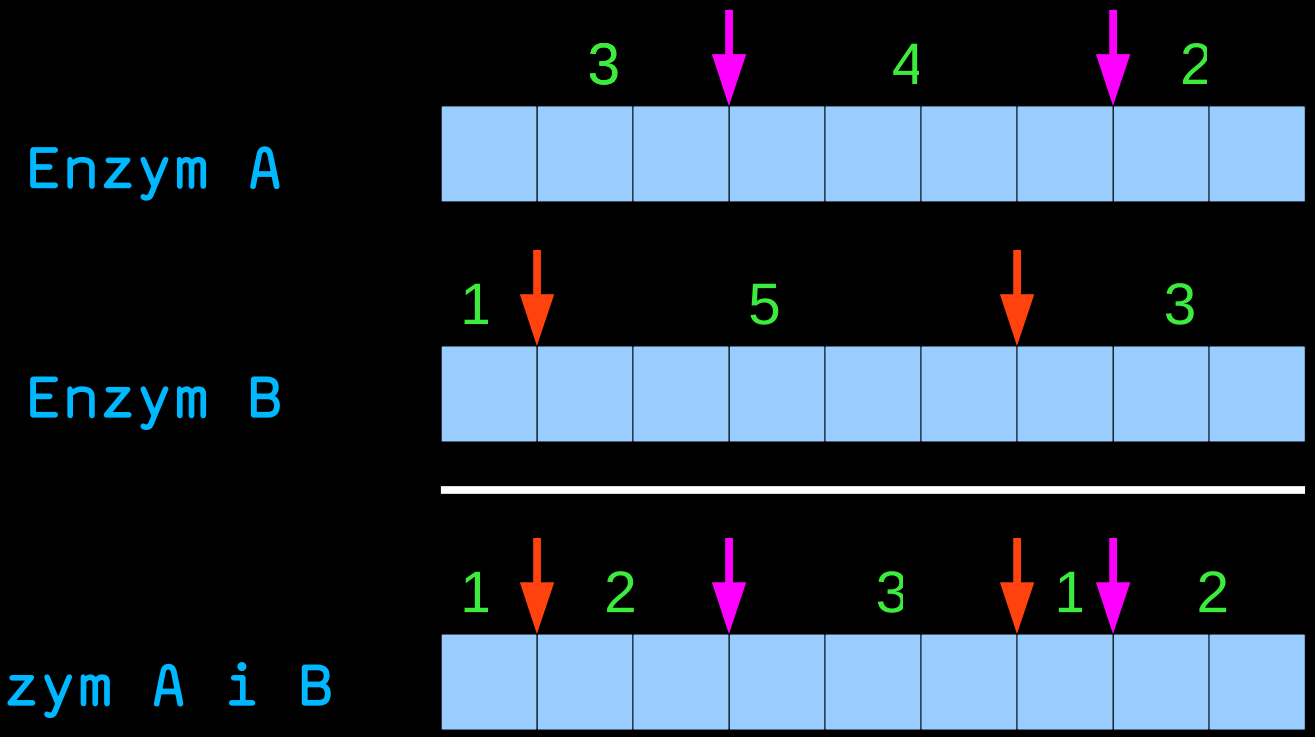
Odečítání aktuálního času

```
from time import time


def run():
  start = time()
  spdp(...)
  end = time()
  return end - start
```

Double digest problem I

- Používáme 2 enzymy A a B
- Tři kopie řetězce necháme plně nasekat: první enzymem A, druhý enzymem B a třetí oběma naráz



Double digest problem II

Vstup:

- Délky úseků řetězce nasekaného enzymem A
- Délky úseků řetězce nasekaného enzymem B
- Délky úseků řetězce nasekaného oběma enzymy

Výstup:

- Seznam pozic, kde řetězec nasekal enzym A
- Seznam pozic, kde řetězec nasekal enzym A (všechny možnosti)

- NP-úplný problém

Double digest problem III

```
# A_ je pole obsahující délky podřetězců řetezce nasekaného enzymem A,
# podobně B_ a AB_, kde AB_ obsahuje hodnoty pro řetězec nasekaný oběma
# enzymy naráz
A_permut  = generate_all_permutations(A_)
B_permut  = generate_all_permutations(B_)
AB_permut = generate_all_permutations(AB_)

# pro každou možnost uspořádání podřetězců nasekaných oběma enzymy
for AB in AB_permut:
  AB_cuts = cuts_from_lengths(AB)        # převeď do formátu pozic seků
  # pro každou možnost uspořádání podřetězců nasekaných enzymem B
  for B in B_permut:
    B_cuts = cuts_from_lengths(B)        # převeď do formátu pozic seků
```

```
# pokud jsou všechny seky enzymem B v množině seků oběma enzymy,
# dává to smysl a pokračujeme dále
if subset(B_cuts,AB_cuts):
  # pro každou možnost uspořádání podřetězců nasekaných enzymem A
  for A in A_permut:
    A_cuts = cuts_from_lengths(A)  # převeď do formátu pozic seků
    # pokud jsou všechny seky enzymem A v množině seků oběma
    # enzymy, dává to smysl a pokračujeme dále
    if subset(A_cuts,AB_cuts):
      # Sjednoť pozice seků enzymů A a B zvlášť do jedné množiny
      M_cuts = merge_cuts(A_cuts,B_cuts)
      # Pokud si tyto dvě množiny odpovídají, jde o možné řešení
      if same_cuts(M_cuts,AB_cuts):
        result.append([A_cuts,B_cuts,M_cuts])
```

Metodika měření

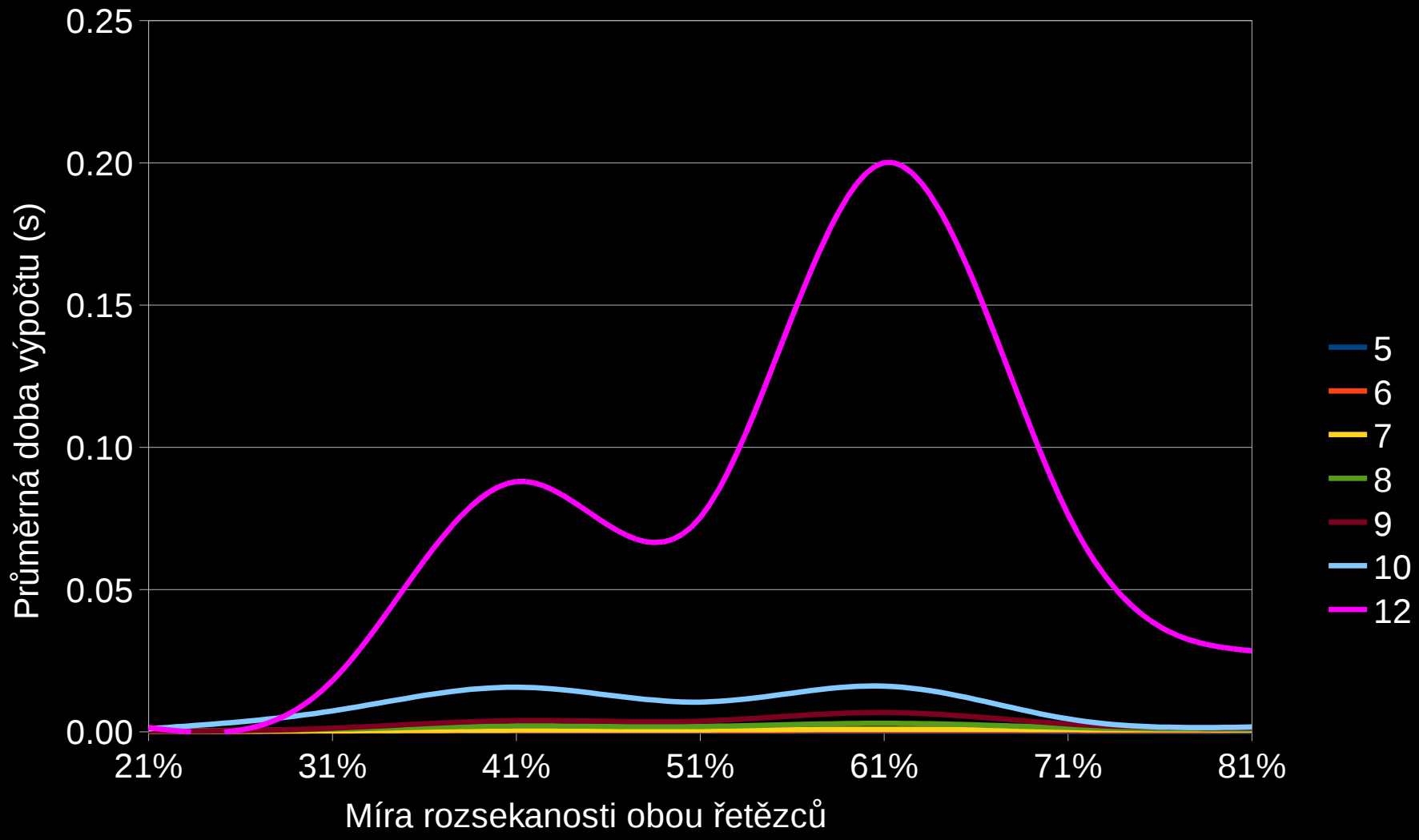
1. Pro všechny délky řetězců:
   2. Pro všechny různé kombinace míry rozsekanosti obou řetězců:
      3. Pro počet testování různých řetězců se stejnou měrou rozsekanosti:
         4. Vygeneruj řetězece rozsekané oběma enzymy danou měrou
         5. Změř čas výpočtu (spuštění několikrát bez zaznamenání výsledků)
         6. Získej výsledky (dalším během jiné implementace algoritmu, která ukládá výsledky)
      7. Zprůměruj a ulož získané výsledky

Metodika měření

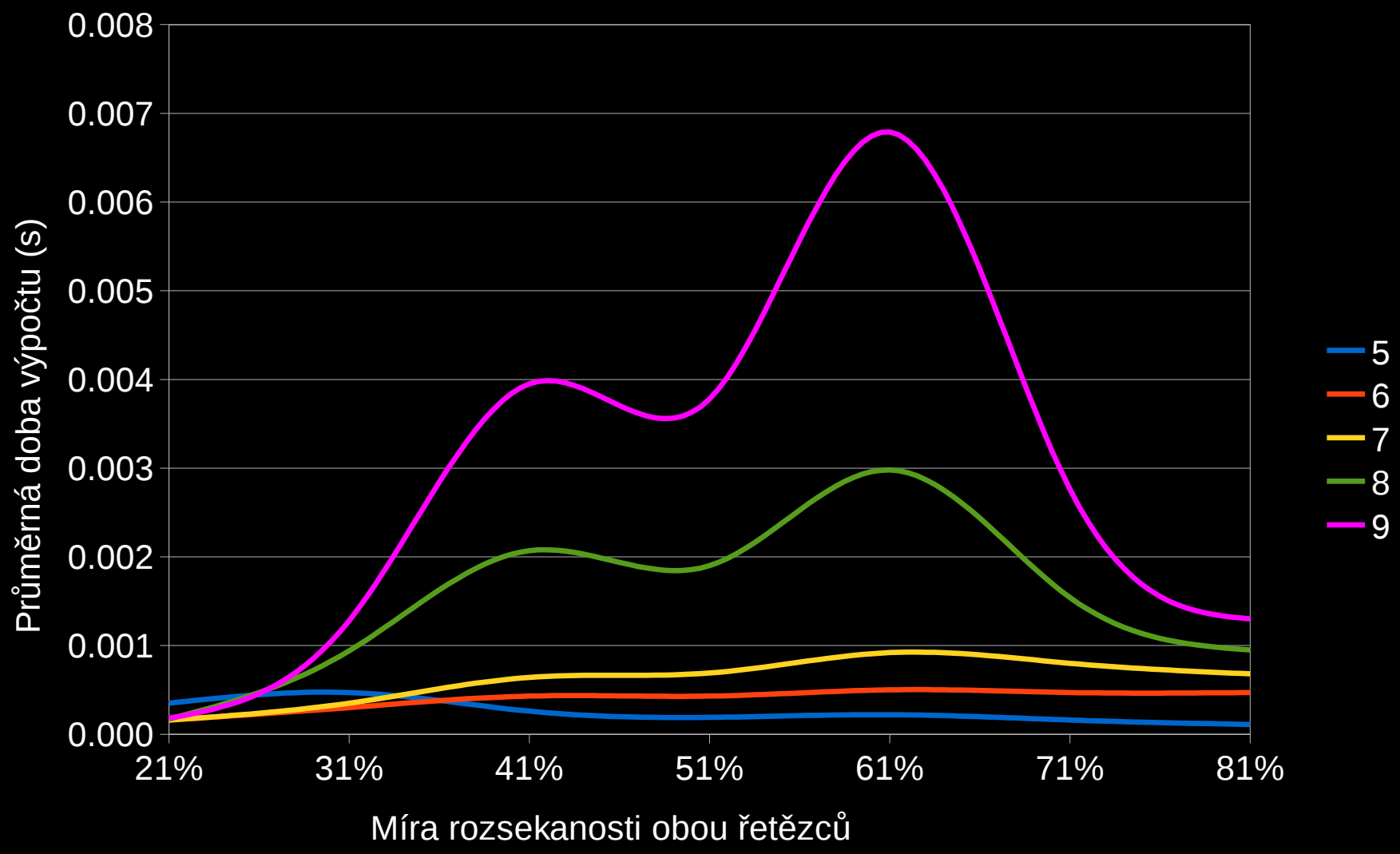
- Testované délky:[5,6,7,8,9,10,12]
- Testované míry rozsekanosti každého řetězce:

  [1%, 21%, 41%, 81%]
- Počet opakování jednoho výpočtu se stejným vstupem: 10
- Počet různých vstupů pro jednu danou míru rozsekanosti: 200

Závislost délky výpočtu na celkové míře rozsekanosti



Závislost délky výpočtu na celkové míře rozsekanosti - detail

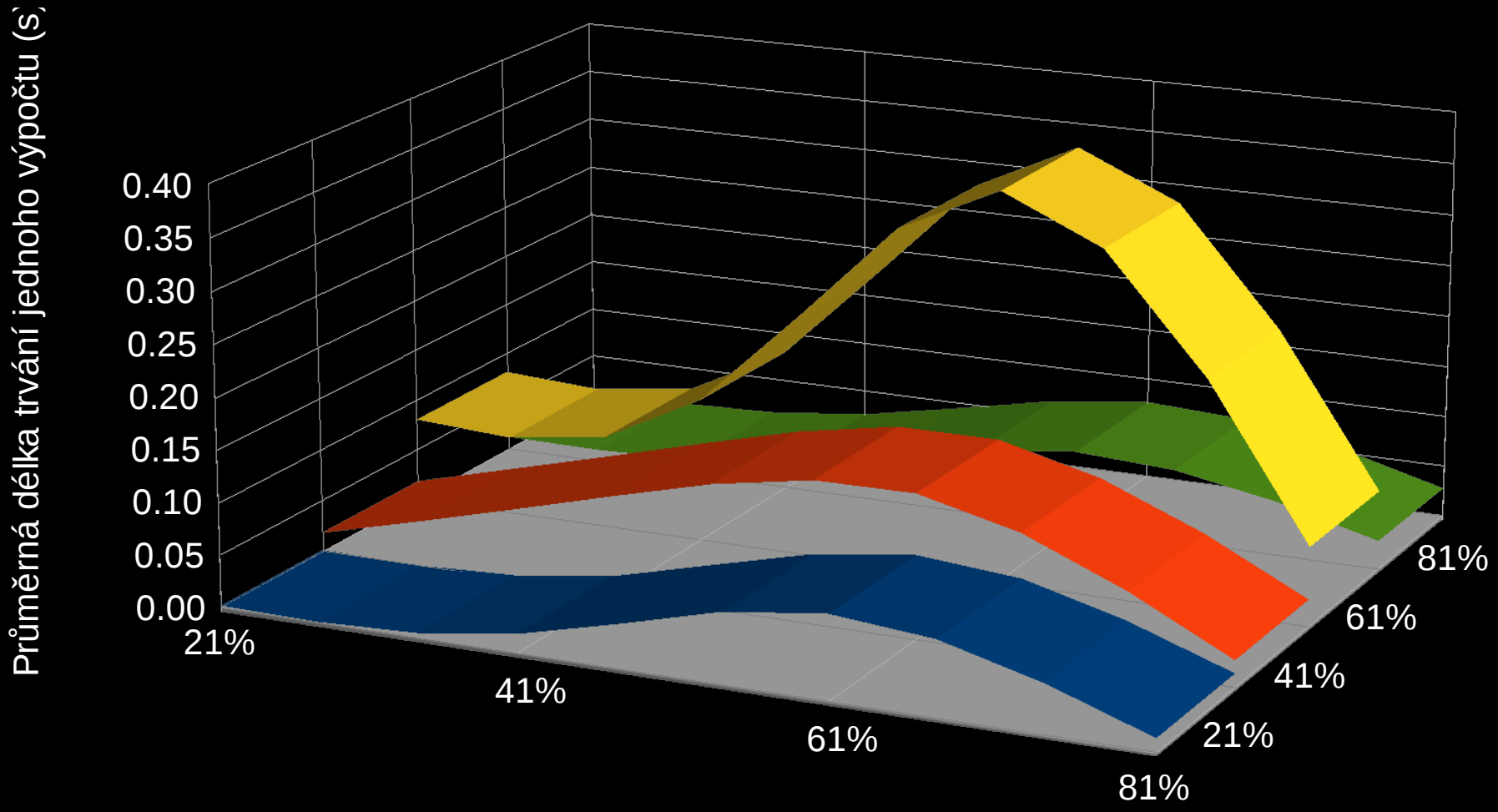


Srovnání délky výpočtu pro různou míru nasekanosti oběma enzymy pro řetězec délky 12

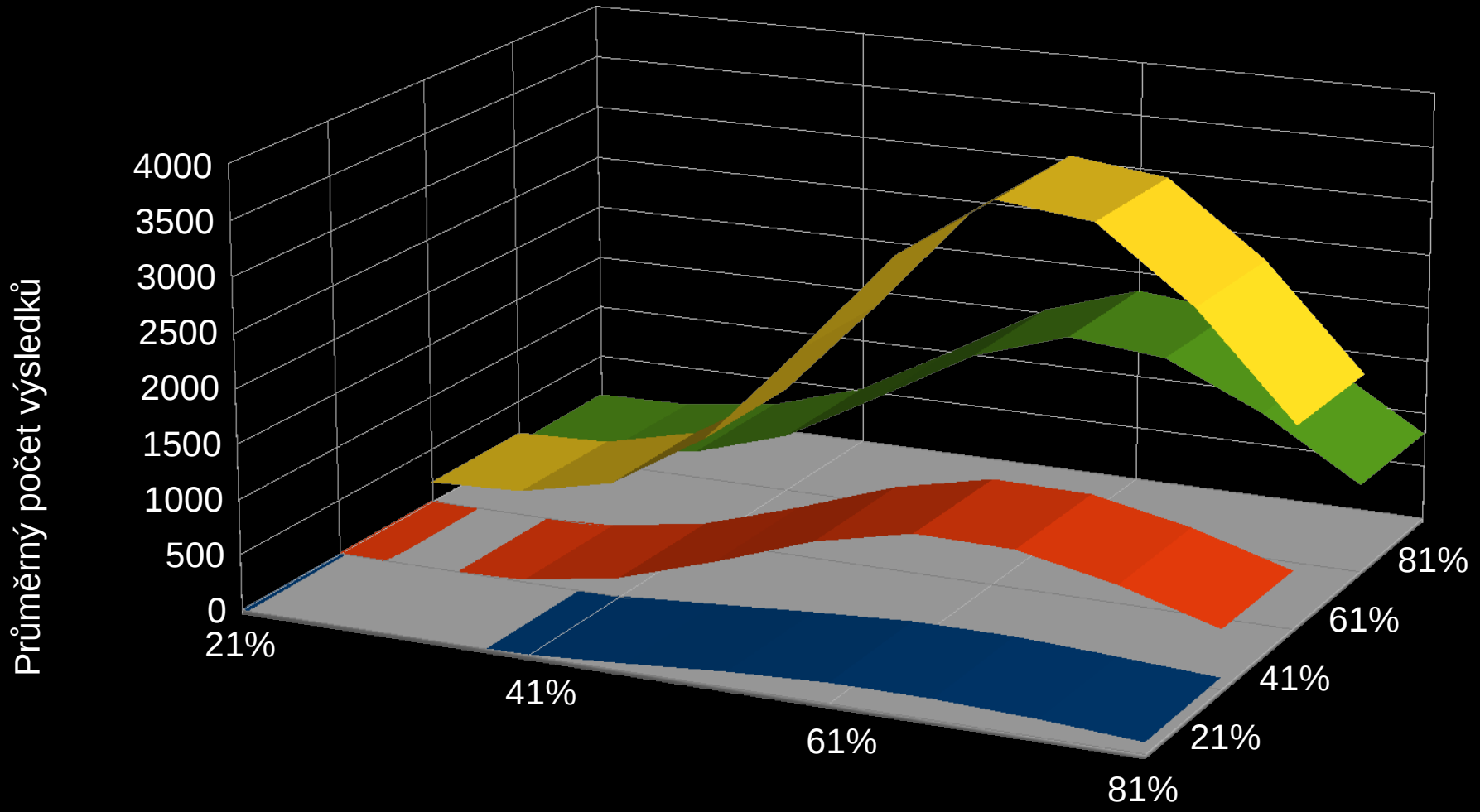
	21%	41%	61%	81%
21%	0.00149	0.01798	0.08078	0.01272
41%	0.01798	0.09511	0.13787	0.02551
61%	0.08078	0.13787	0.37464	0.07640
81%	0.01272	0.02551	0.07640	0.02846

Výsledky jsou v sekundách

Srovnání délky výpočtu pro různou míru nasekanosti oběma enzymy pro řetězec délky 12

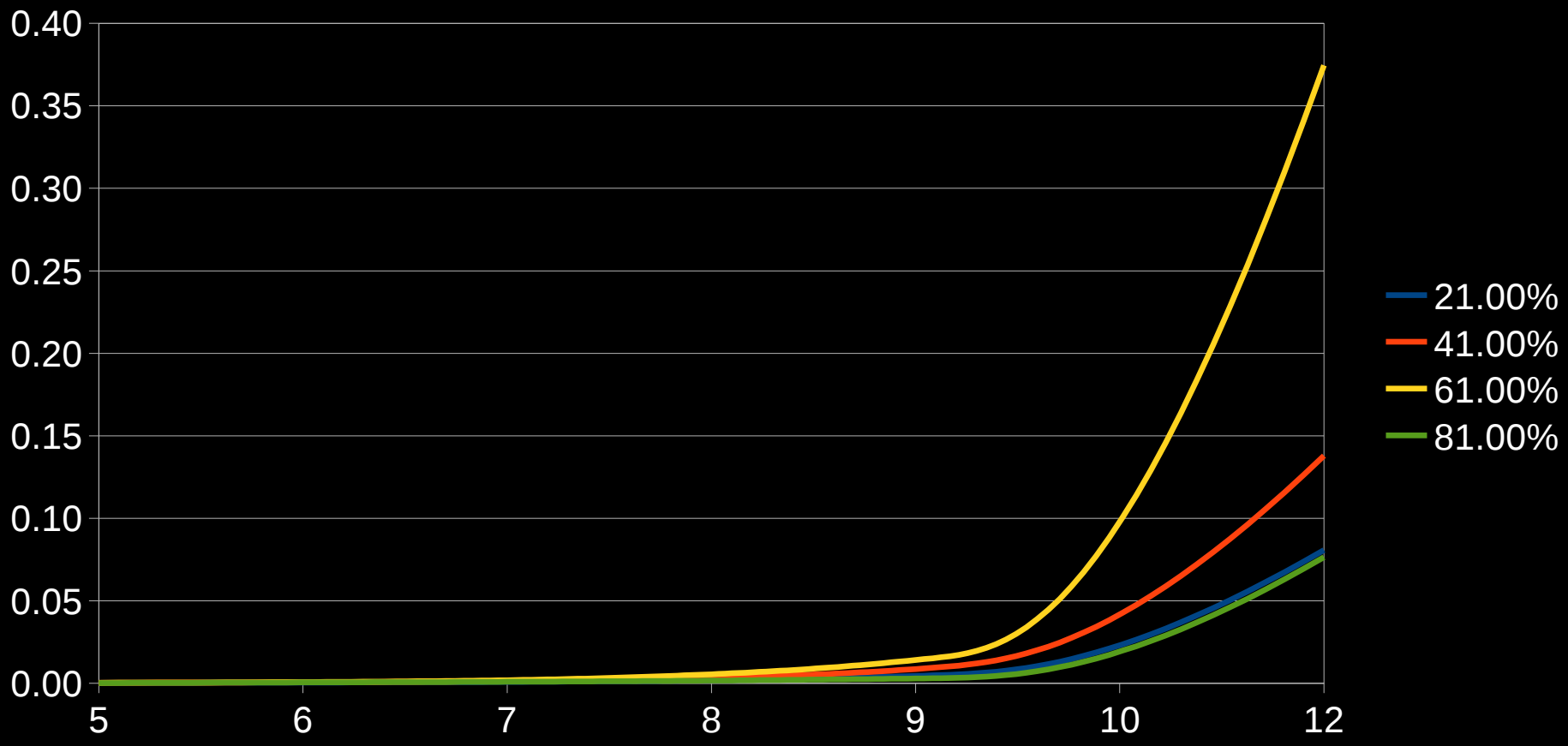


Srovnání počtu možných výsledků pro různou míru nasekanosti oběma enzymy pro řetězec délky 12

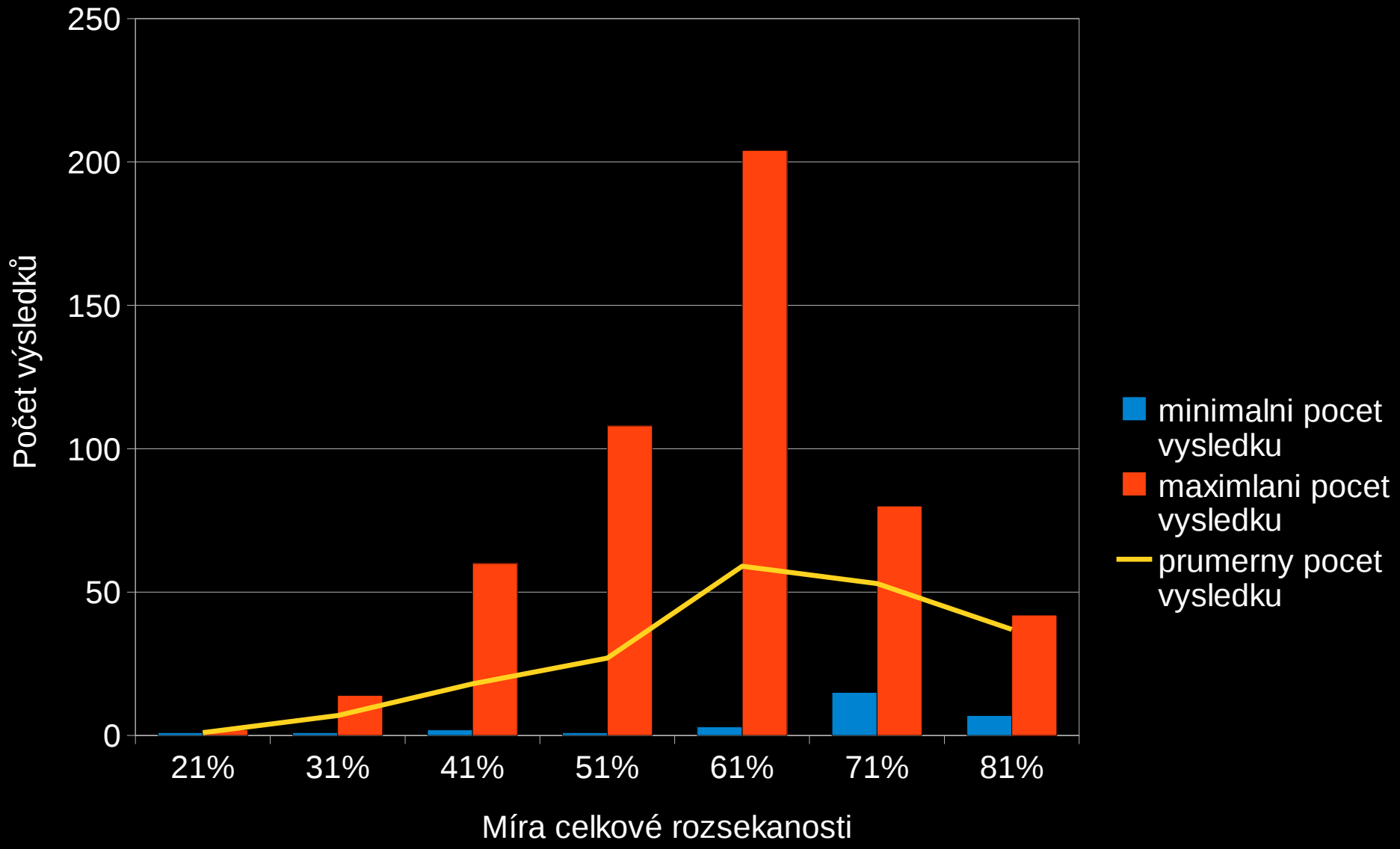


Délka výpočtu (ve vteřinách)

První řetězec je fixně rozsekán v 61% možných míst, mění se délka řetězce a míra nasekanosti druhého řetězce.

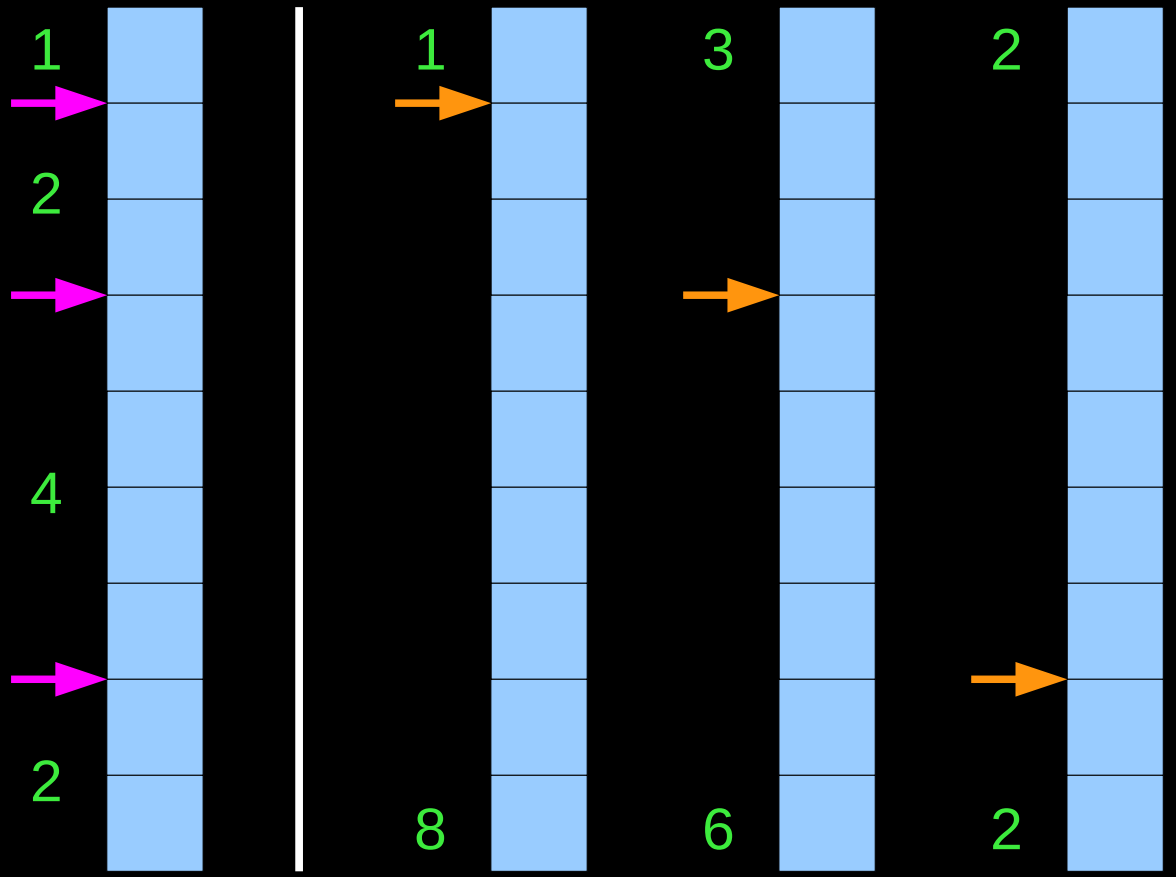


Počty výsledků



Simplified partial digest I

- Používáme 1 enzym A
- Několik kopií řetězce necháme nasekat: první úplně, ostatní každý pouze na jednom místě



Simplified partial digest II

Vstup:

- Množina délek úseků řetězce nasekaného úplně
- Množina dvojic délek úseků řetězců přeseklých pouze jednou

Výstup:

- Seznam pozic, kde enzym nasekal řetězec

(všechny možnosti)

Simplified partial digest III

- Každá dvojice úseků z druhého experimentu určuje dvě potenciální dělící místa
- Seřadím je podle vzdálenosti od levého okraje
- Začínám s prázdným řešením a i = 0

SPDP(reseni,i)

- Pokud uz je reseni cele, tak ho vypisu a vynorim se z rekurze
- Jinak vezmu ze seznamu i-té dělící místo a zkusím ho přidat k řešení

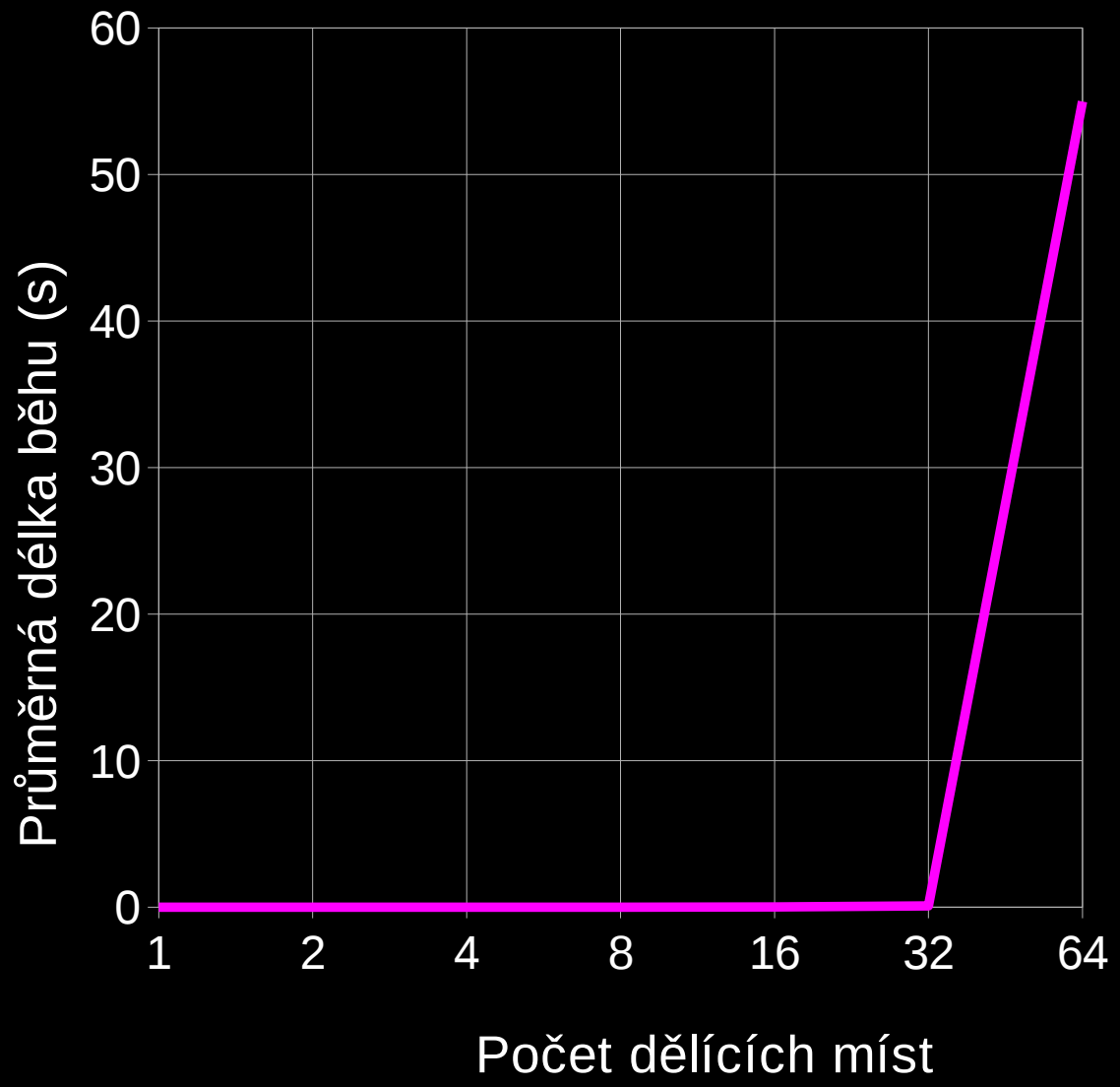
Simplified partial digest IV

- Tím mi vzniknou dva nové úseky, levý musí být součástí výsledku prvního experimentu (druhý budu ještě v dalších krocích sekat)
- Pokud můžu dělící místo přidat, tak upravím aktuální řešení a pokračuji rekurzí na nové řešení a další dělící místo
- Po vynoření z rekurze, nebo pokud nemůžu dělící místo přidat, tak pokračuji rekurzí na neupravené řešení a na další dělící místo

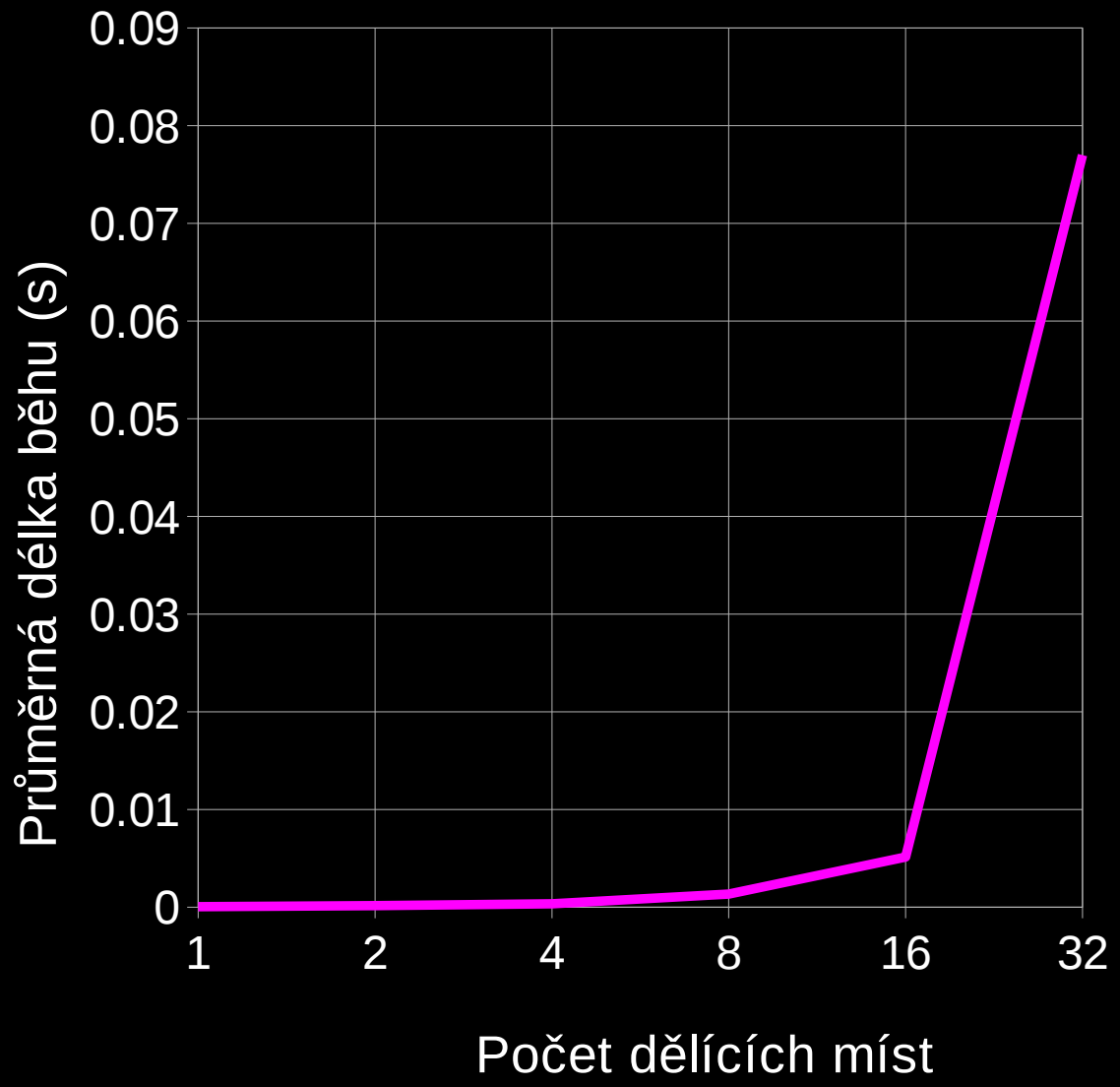
Metodika měření

```
Pro všechny počty dělících míst (1 - 30):
    Pro všechna procenta opakování délky fragmentu
    (0%  - 100% po 10%):
        Pro počet opakování měření (100):
            Vygeneruj zadání
            Spusť algoritmus a zaznamenej dobu běhu
        Zprůměruj dobu běhu a ulož výsledky
```

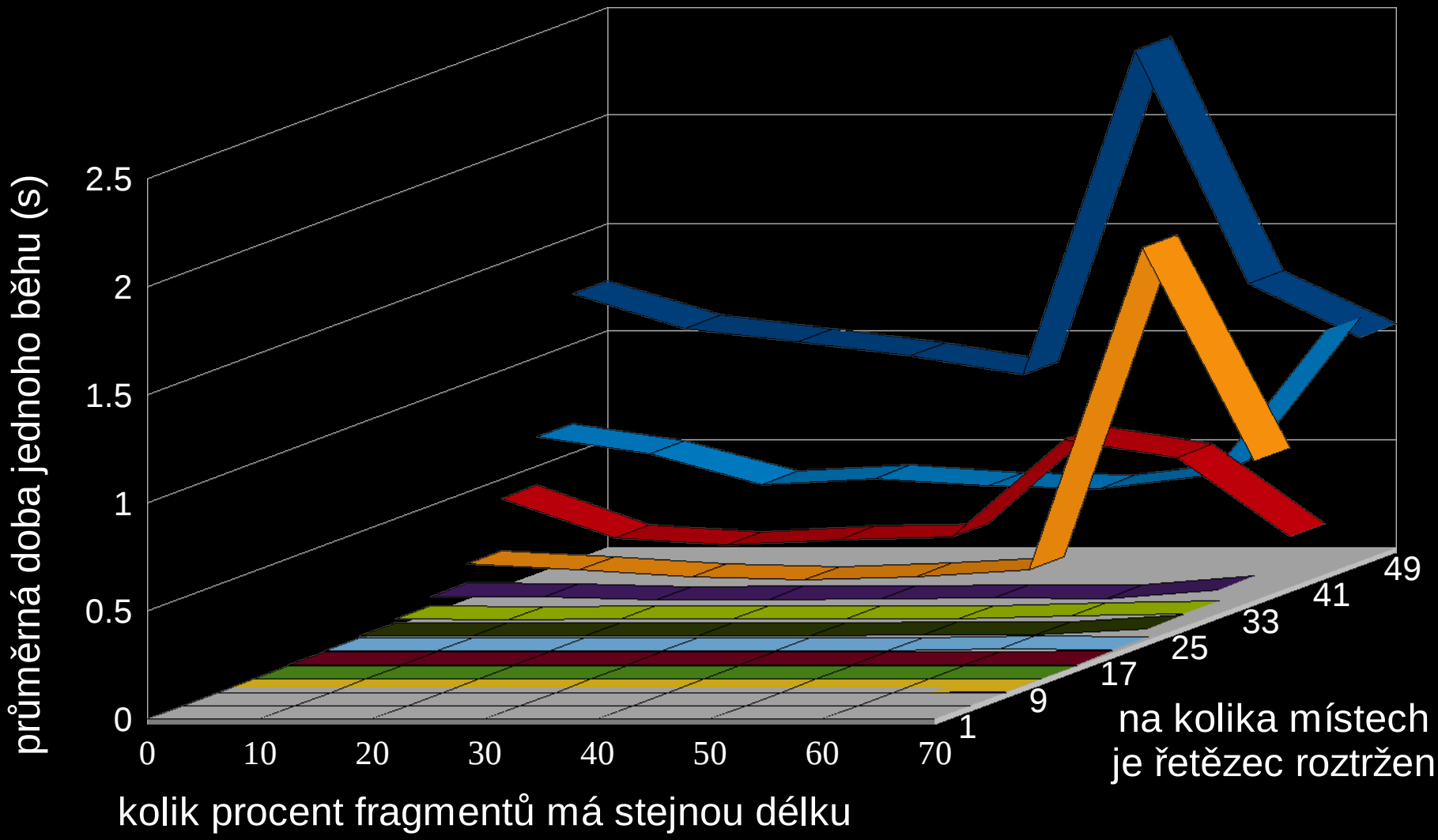
Závislost délky běhu na počtu dělících míst



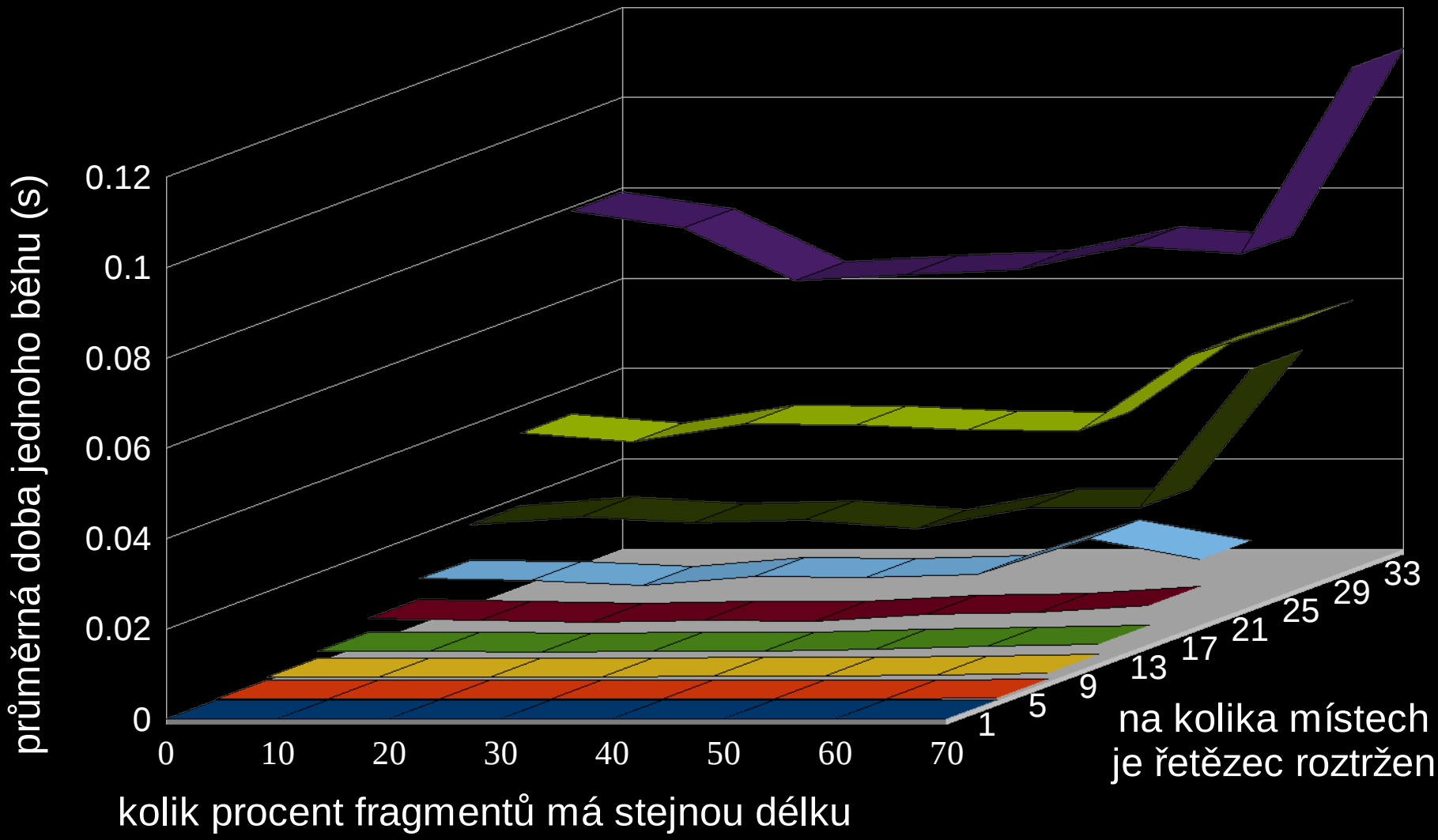
Závislost délky běhu na počtu dělících míst - detail



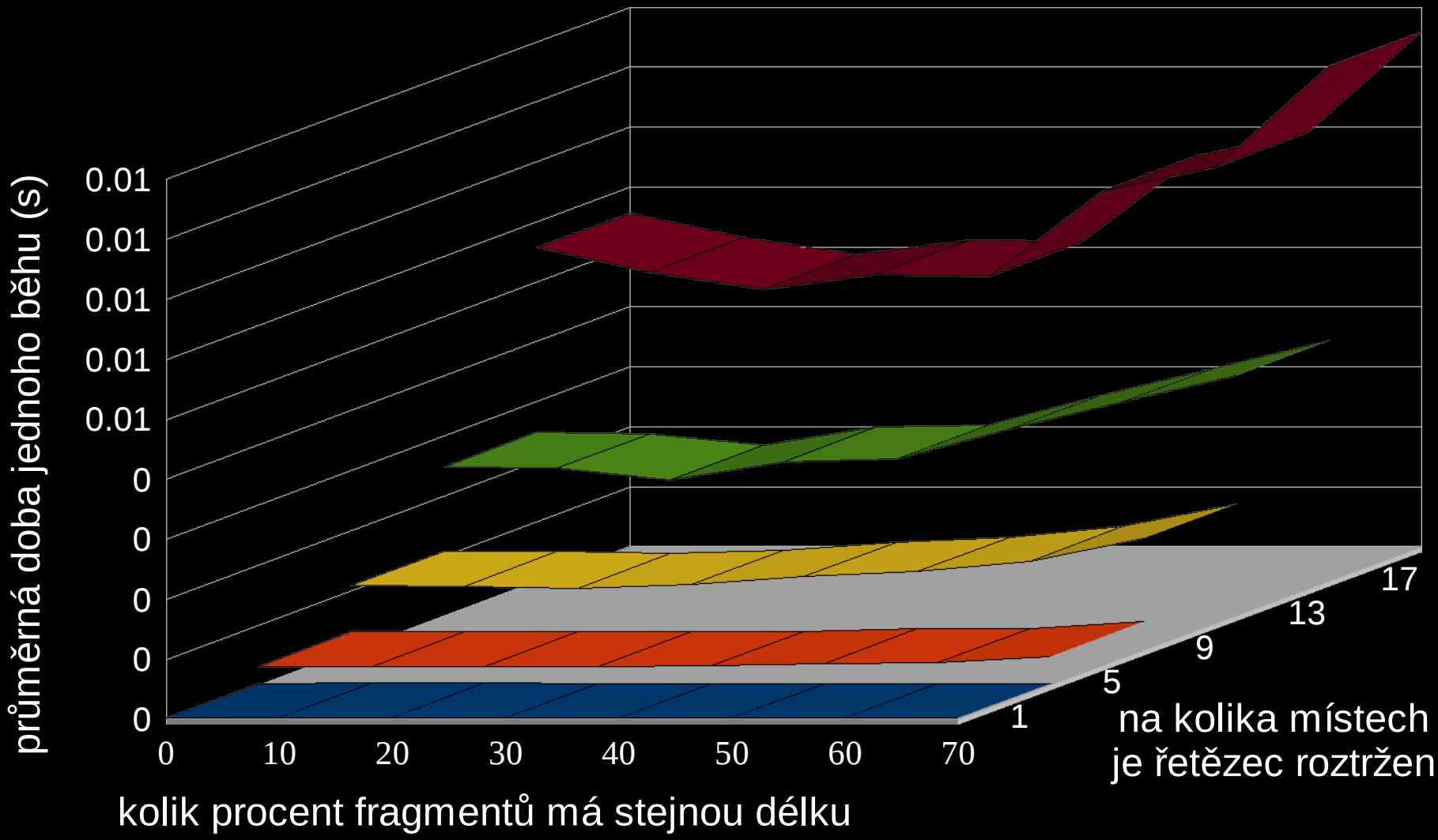
Závislost doby běhu na míře rozsekanosti a množství stejně dlouhých řetězců



Závislost doby běhu na míře rozsekanosti a množství stejně dlouhých řetězců - detail



Závislost doby běhu na míře rozsekanosti a množství stejně dlouhých řetězců - detail



Konec, děkujeme