Porovnání – vyhledání silně souvislých komponent

Gabowův algoritmus a Tarjanův algoritmus

Jan Wrona Tomáš Varga

Fakulta informačních technologií
Vysoké učení technické v Brně

Grafové algoritmy, 2014

Obsah

Gabowův algoritmus

Algorithm 1 Funkce GABOW-VISIT(v).

```
preorder[v] ← c;  c ← c + 1;  push(S, v);  push(P, v)
for w in Adj[v] do
    if preorder[w] = ∞ then
        GABOW-VISIT(w)
    else if w is in S then
        repeat
            pop(P)
        until preorder[peak(P)] ≤ preorder[w]
    end if
end for

if v = peak(P) then
    vytvoř novou silně souvislou komponentu c
    repeat
        w ← pop(S)
        přidej w do komponeny c
    until v = w
    pop(P)
    return c
end if
```

Tarjanův algoritmus

Algorithm 2 Funkce TARJAN-VISIT(v).

$index[v] \leftarrow index$; $lowlink[v] \leftarrow index$; $index \leftarrow index + 1$; $push(S, v)$
for w in $Adj[v]$ **do**
  if $index[w] = \infty$ **then**
    TARJAN-VISIT(w)
  else if w is in S **then**
    $lowlink[v] \leftarrow min(lowlink[v], index[w])$
  end if
end for

if $lowlink[v] = index[v]$ **then**
  vytvoř novou silně souvislou komponentu c
  repeat
    $w \leftarrow pop(S)$
    přidej w do komponeny c
  until $v = w$
  return c
end if

Asymptotická časová složitost

- Reprezentace seznamem sousedů.

- **Gabowův** a **Tarjanův**: upravené DFS. První průchod $O(m+n)$, druhý průchod $O(n)$.
- **Tarjanův** s **Nuutilovou** modifikací: upravený pouze druhý průchod, zlepšení složitosti pro řídké grafy.
- **Kosarajův**: dva kompletní průchody grafem, transpozice, složitost $\Theta(m+n)$.

Obsah

Implementace

- Python2.
- Knihovny *NetworkX*, *psutil*.
- *NetworkX* nepřipouští smyčky v orientovaném grafu.

- **Gabowův** a **Tarjanův**: vlastní implementace i reprezentace grafu.
- **Tarjanův** s **Nuutilovou** modifikací a **Kosarajův**: součást knihovny NetworkX.

Návrh testů

Algorithm 3 Metoda měření.

```
for i ← 1 to repetitions do
    startTime ← cpuTime
    startMemory ← memory
    for j ← 1 to multiplication do
        SCC-ALGORITHM(G)
    end for
    totalTime ← cpuTime − startTime
    totalMemory ← memory − startMemory
end for
```

- Vnější cyklus zmenšuje chybu měření – výběr minima z naměřených časů.
- Vnitřní cyklus eliminuje nízké rozlišení měřeného procesorového času.

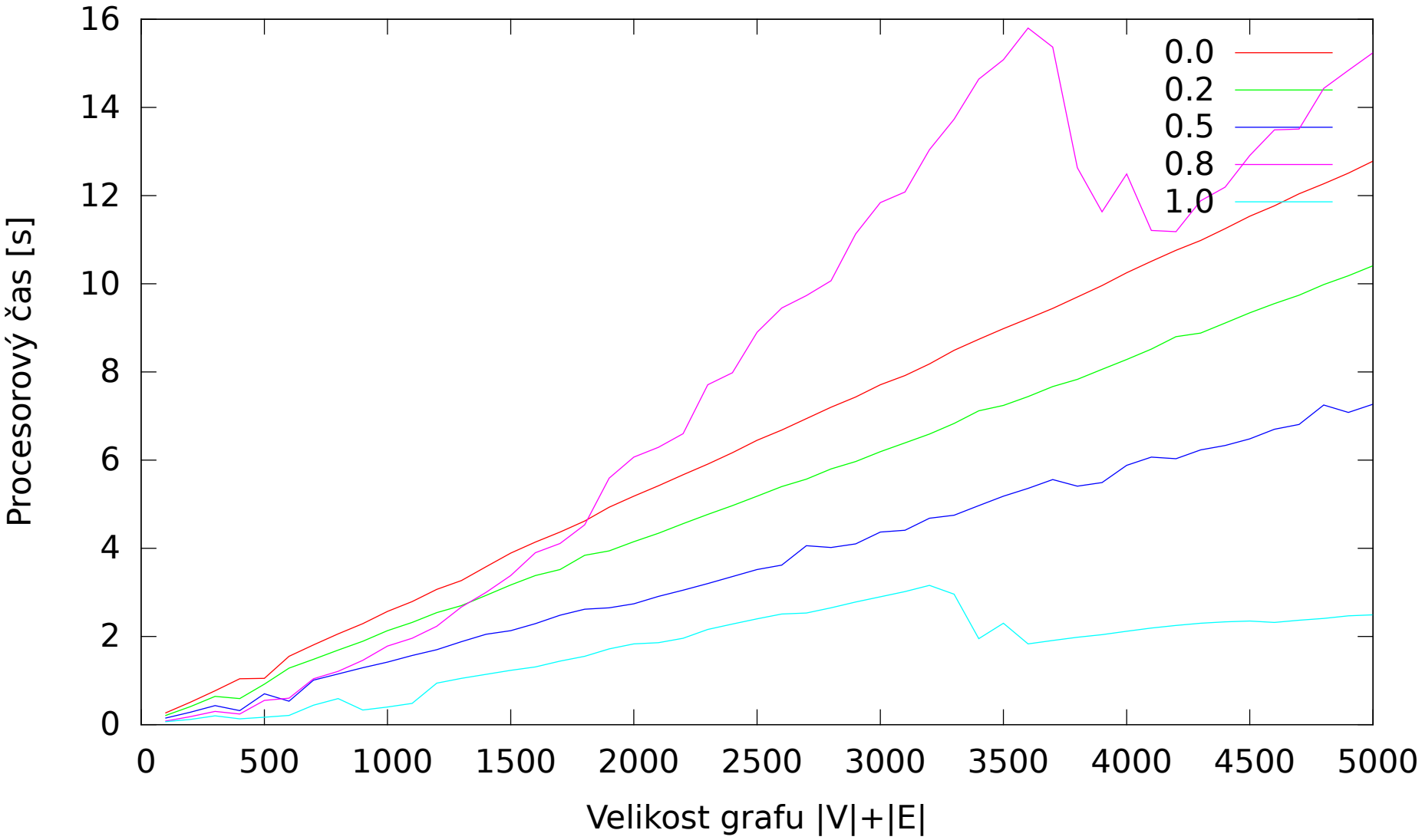
Způsob měření času a paměti

- Čas měřen pomocí `time.clock()`. Jedná se o procesorový čas v sekundách.
- Paměť měřena pomocí metody `memory_info()` třídy `Process` knihovny *psutil*.
- `memory_info()` zjistí velikost fyzické paměti využité procesem – resident set size.

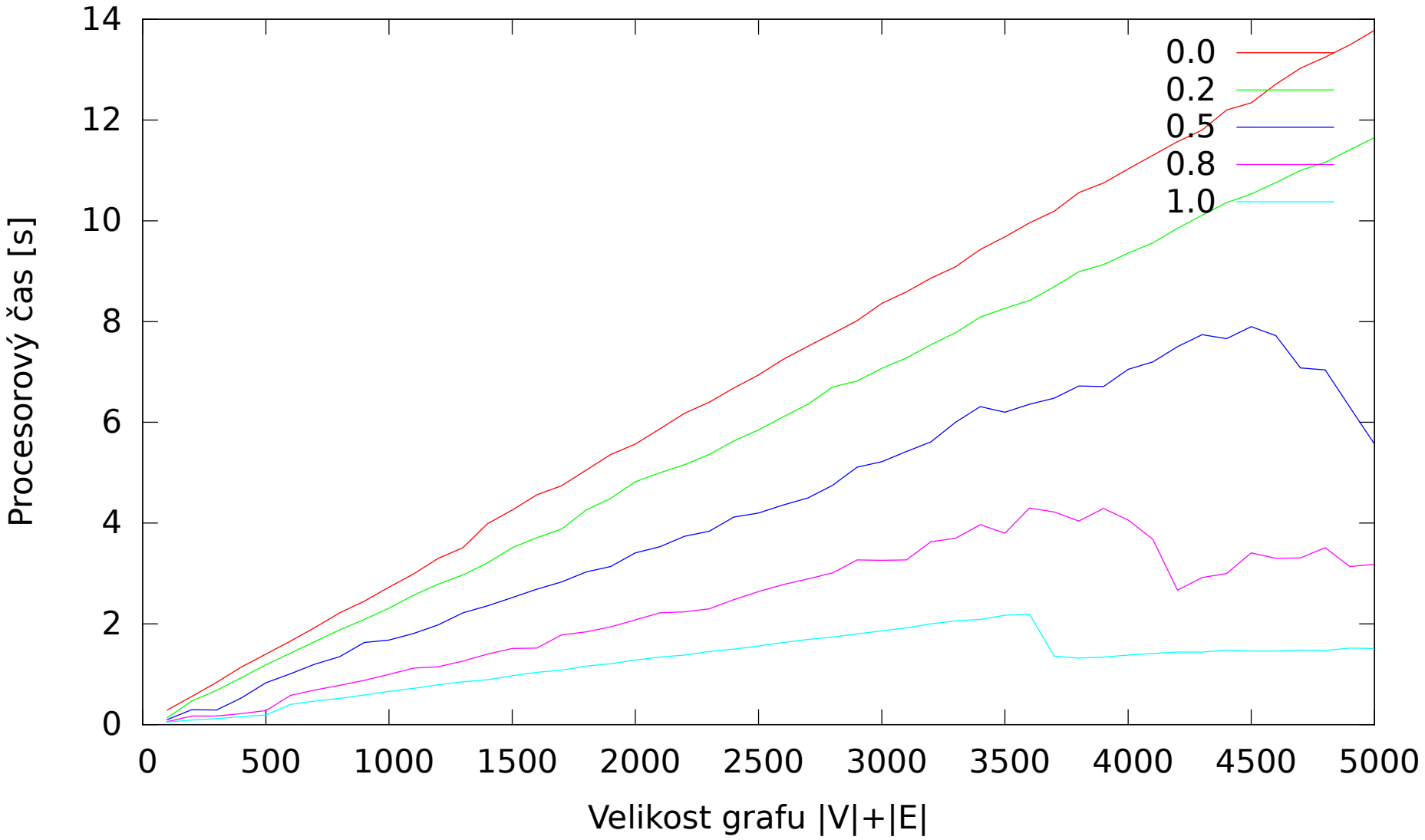
Obsah

Table of Contents

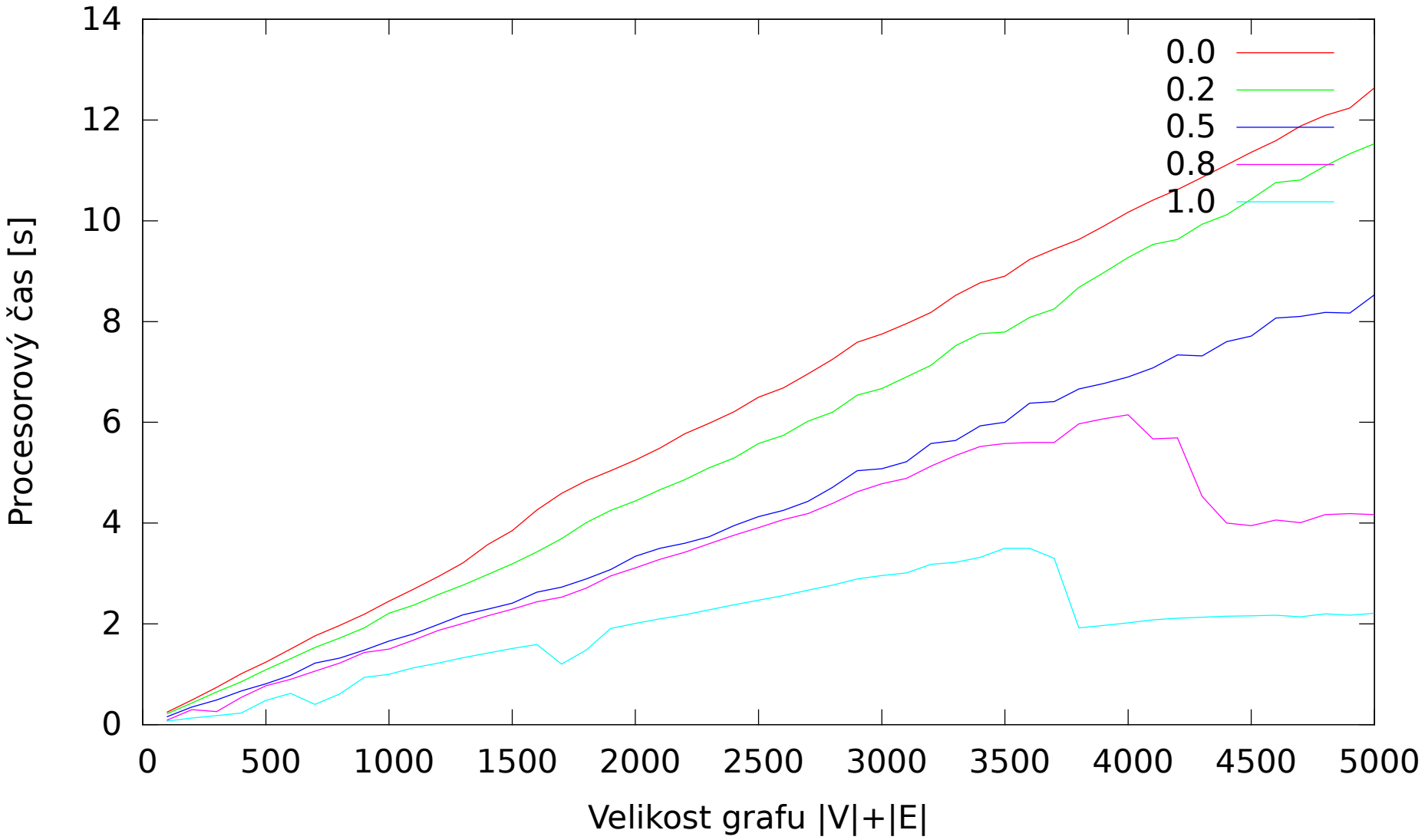
Gabowův algoritmus



Tarjanův algoritmus



Tarjanův algoritmus s Nuutilovou modifikací



Kosarajův algoritmus

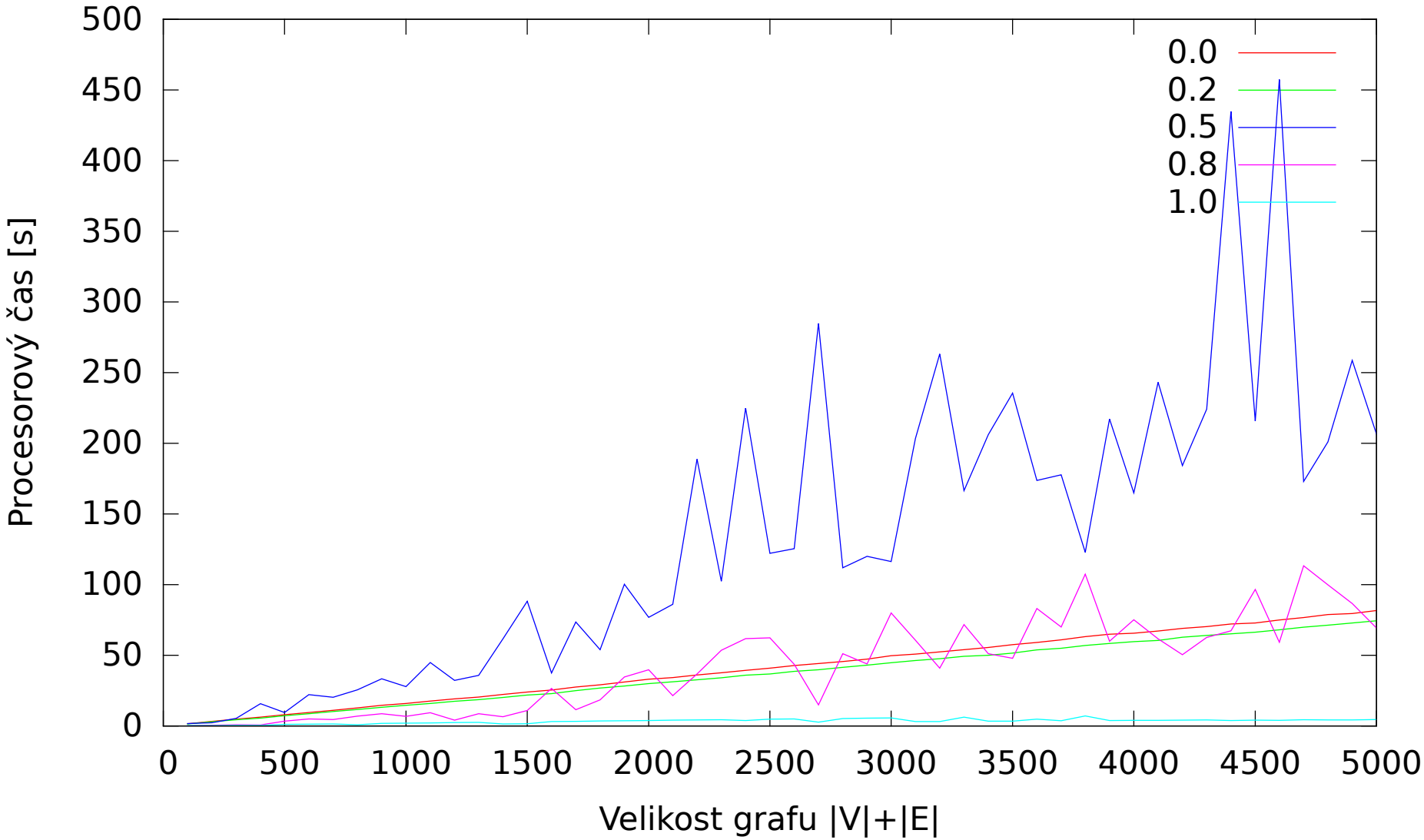
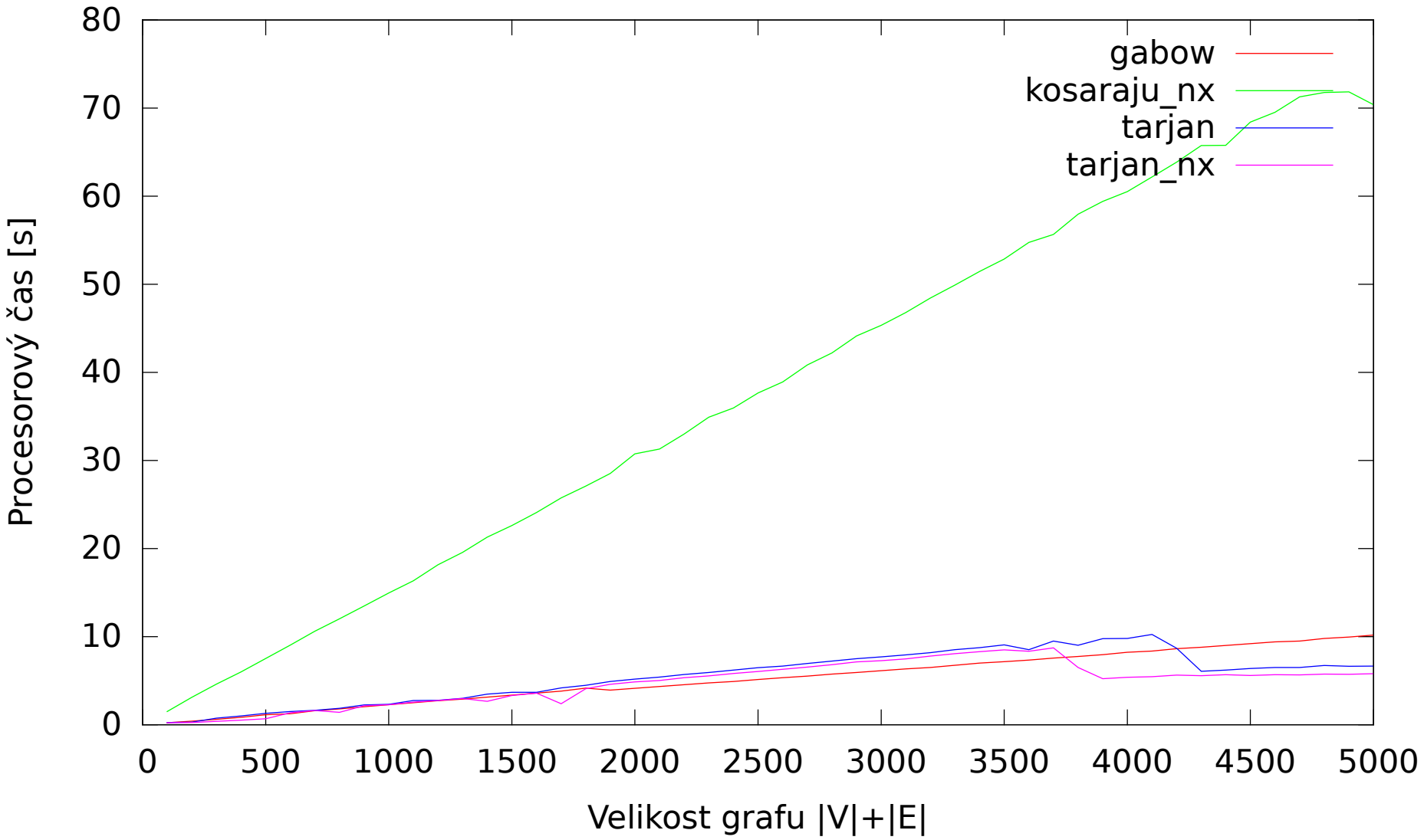
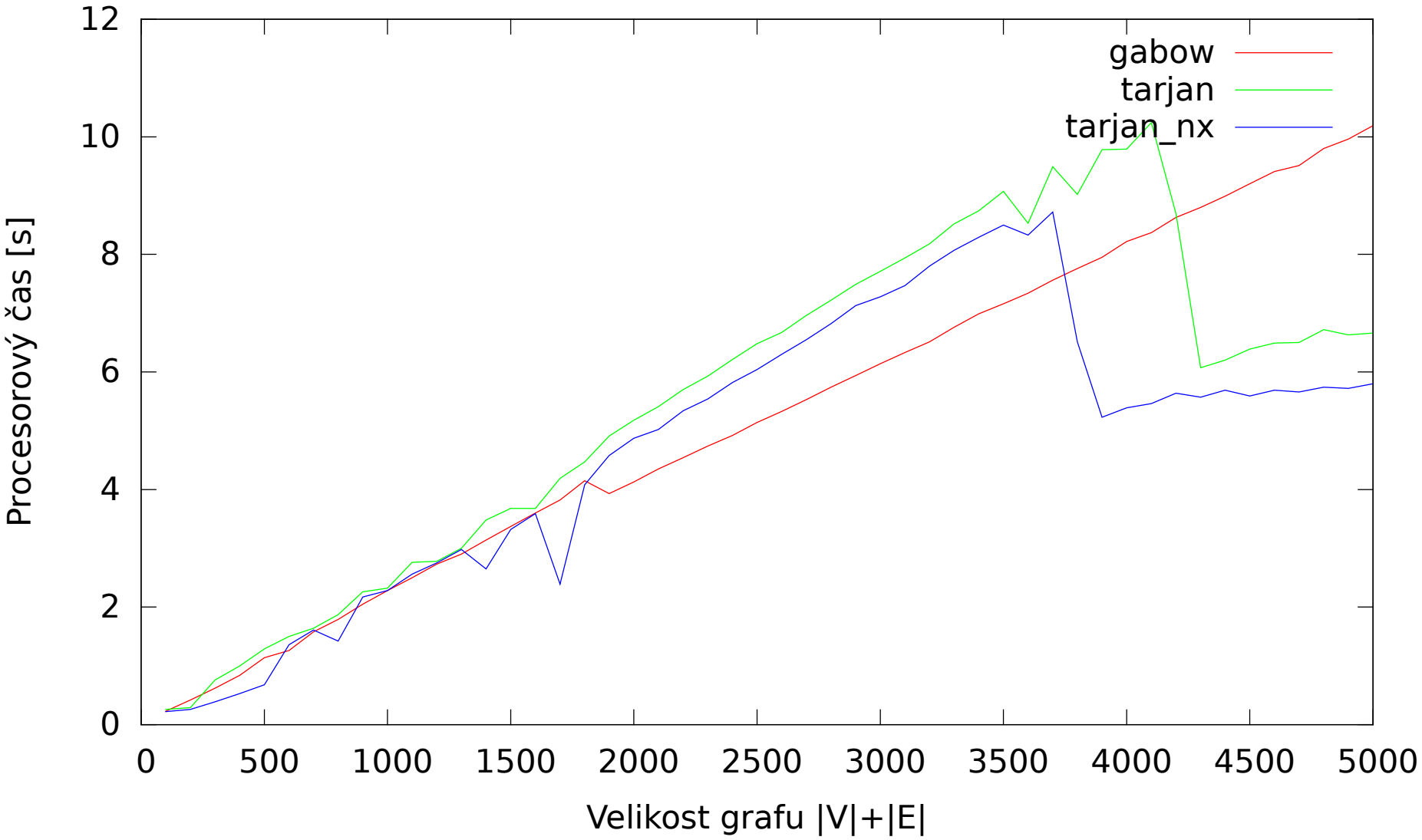


Table of Contents

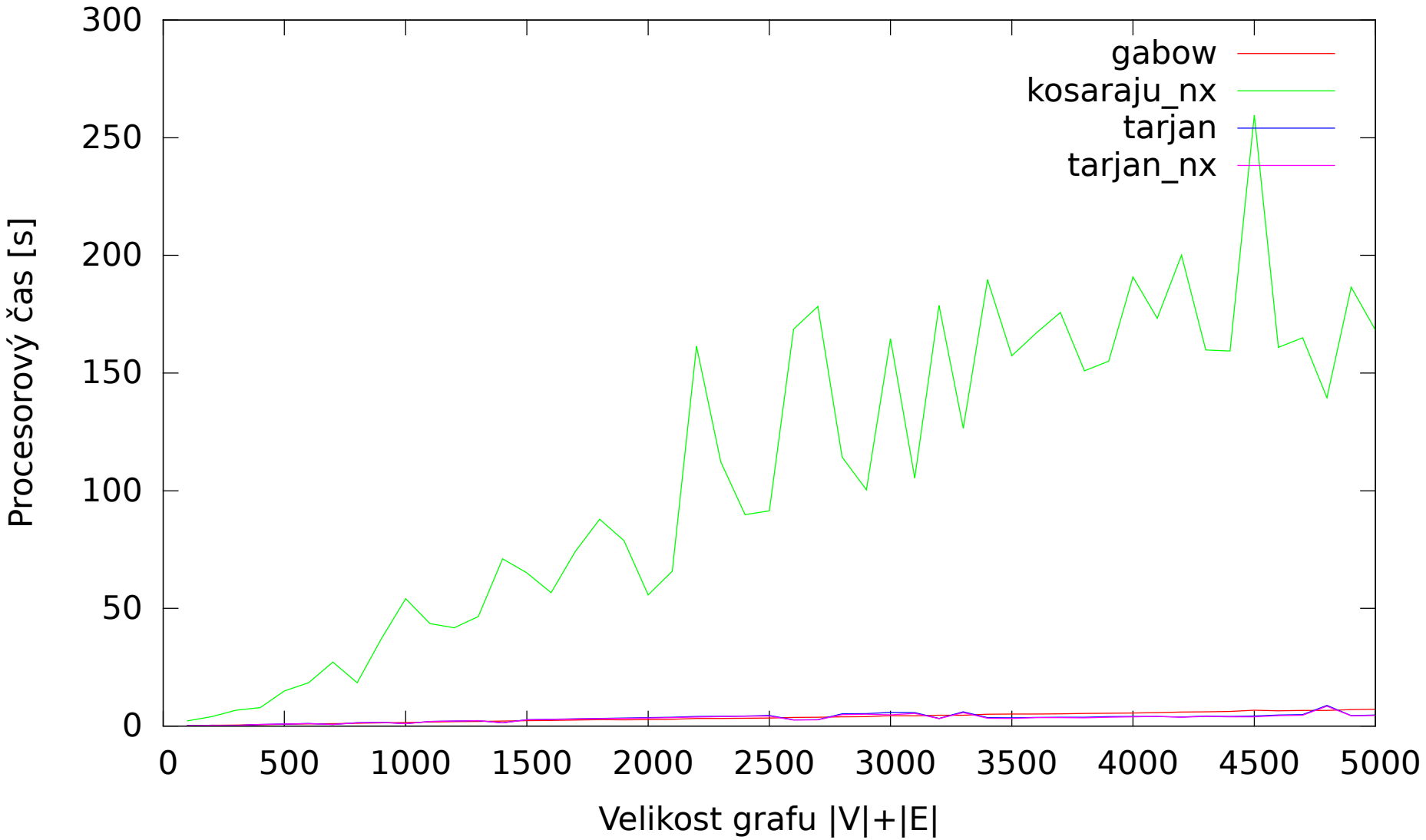
Srovnání – řídký graf



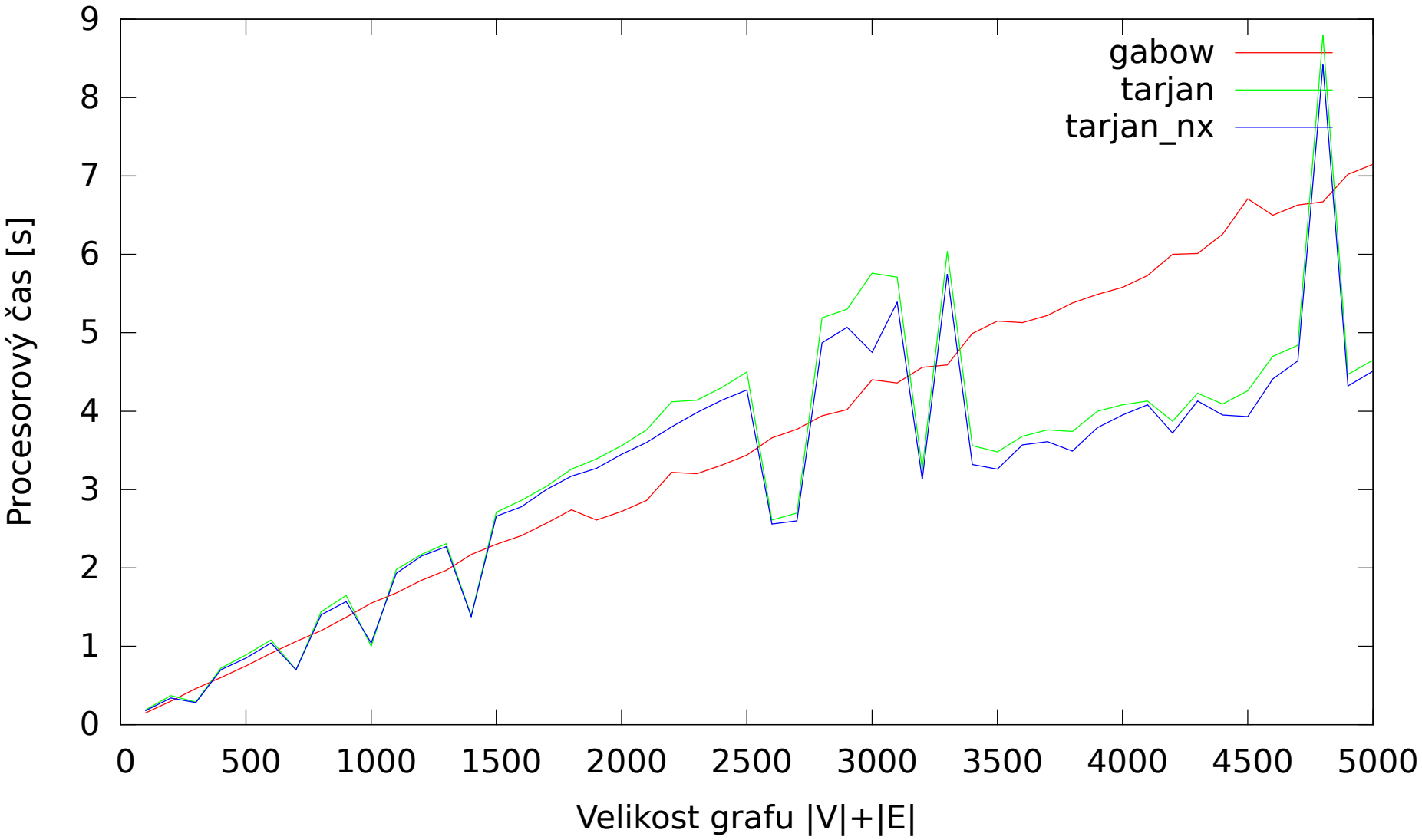
Srovnání – řídký graf



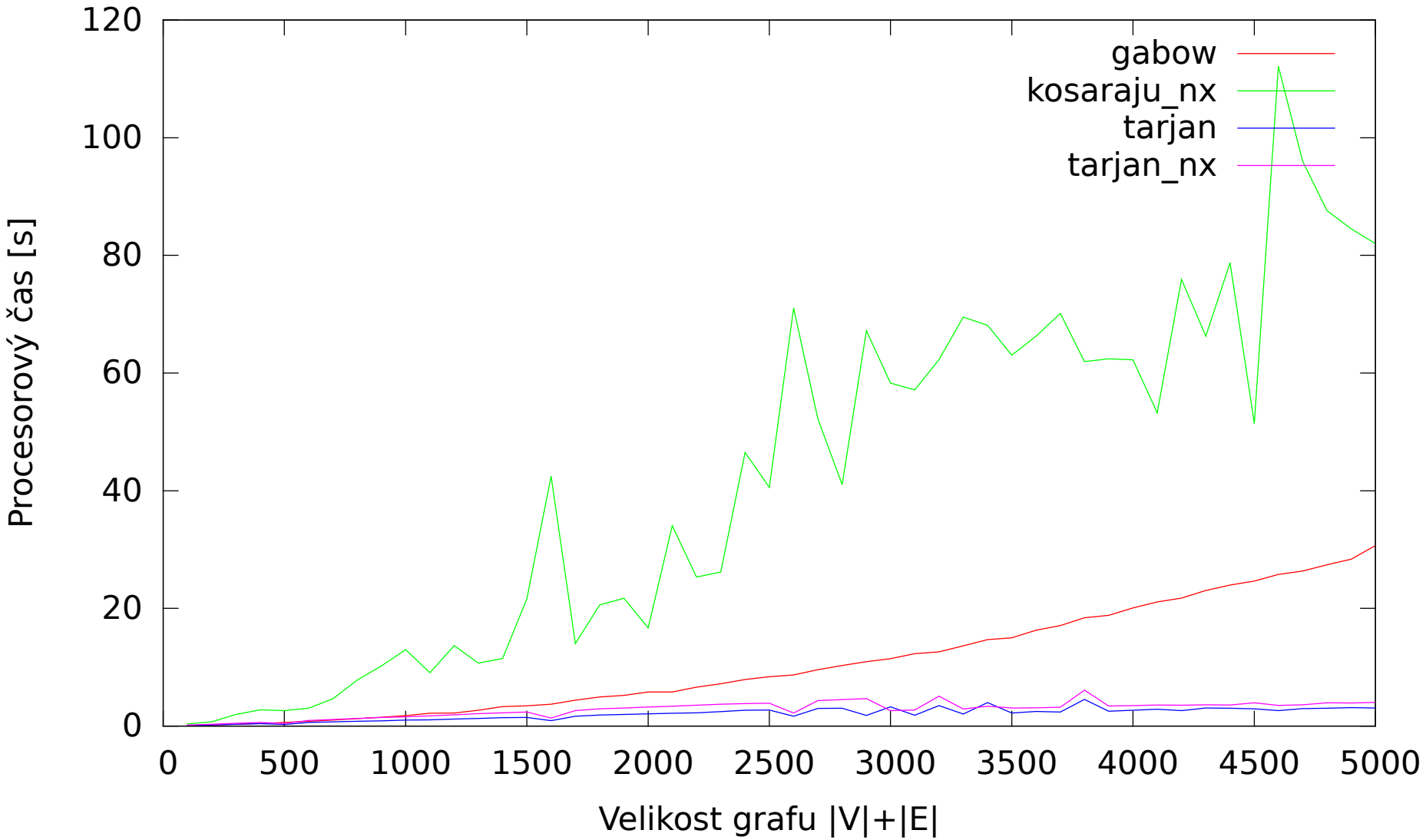
Srovnání – vyvážený graf



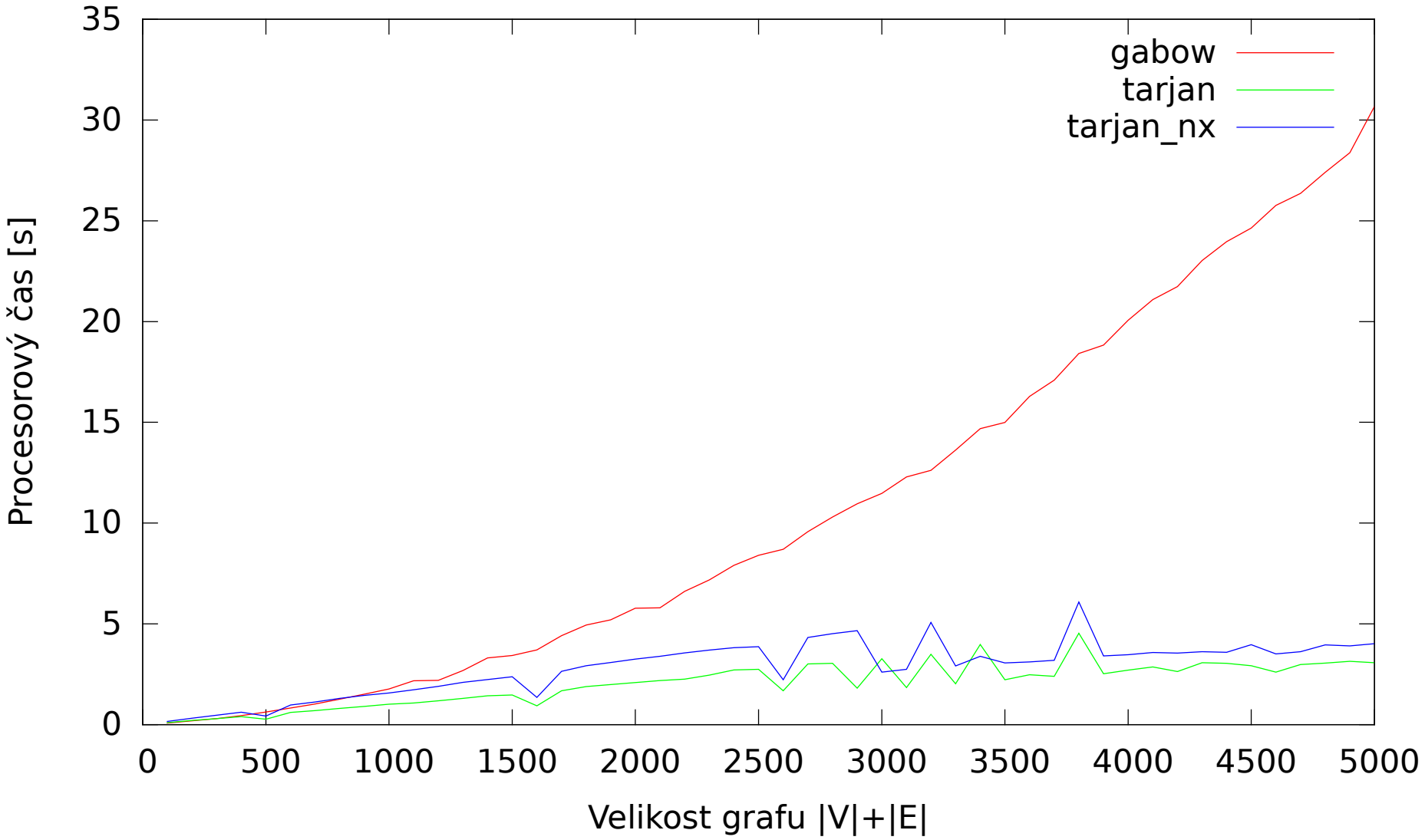
Srovnání – vyvážený graf



Srovnání – hustý graf



Srovnání – hustý graf



Srovnání – úplný graf

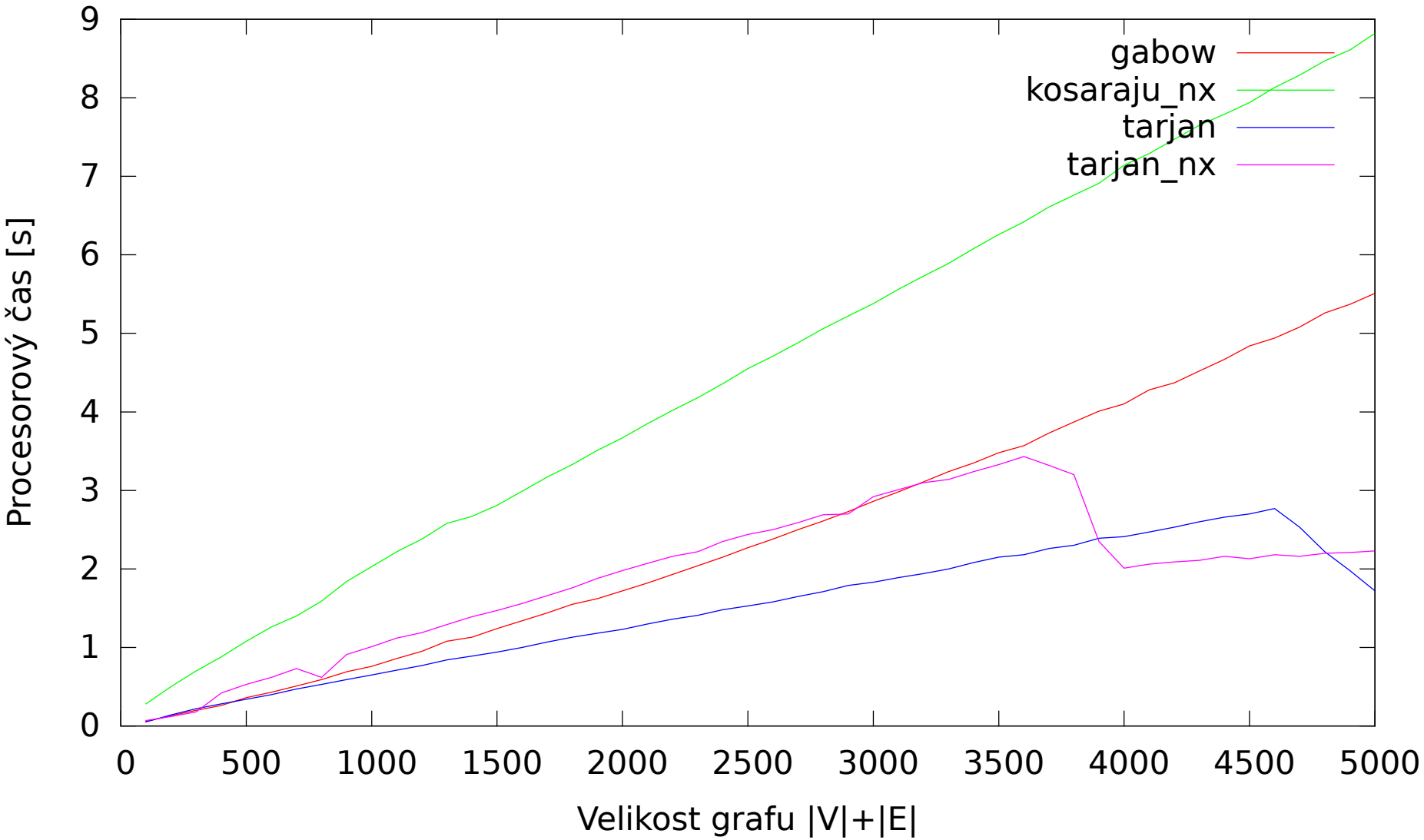
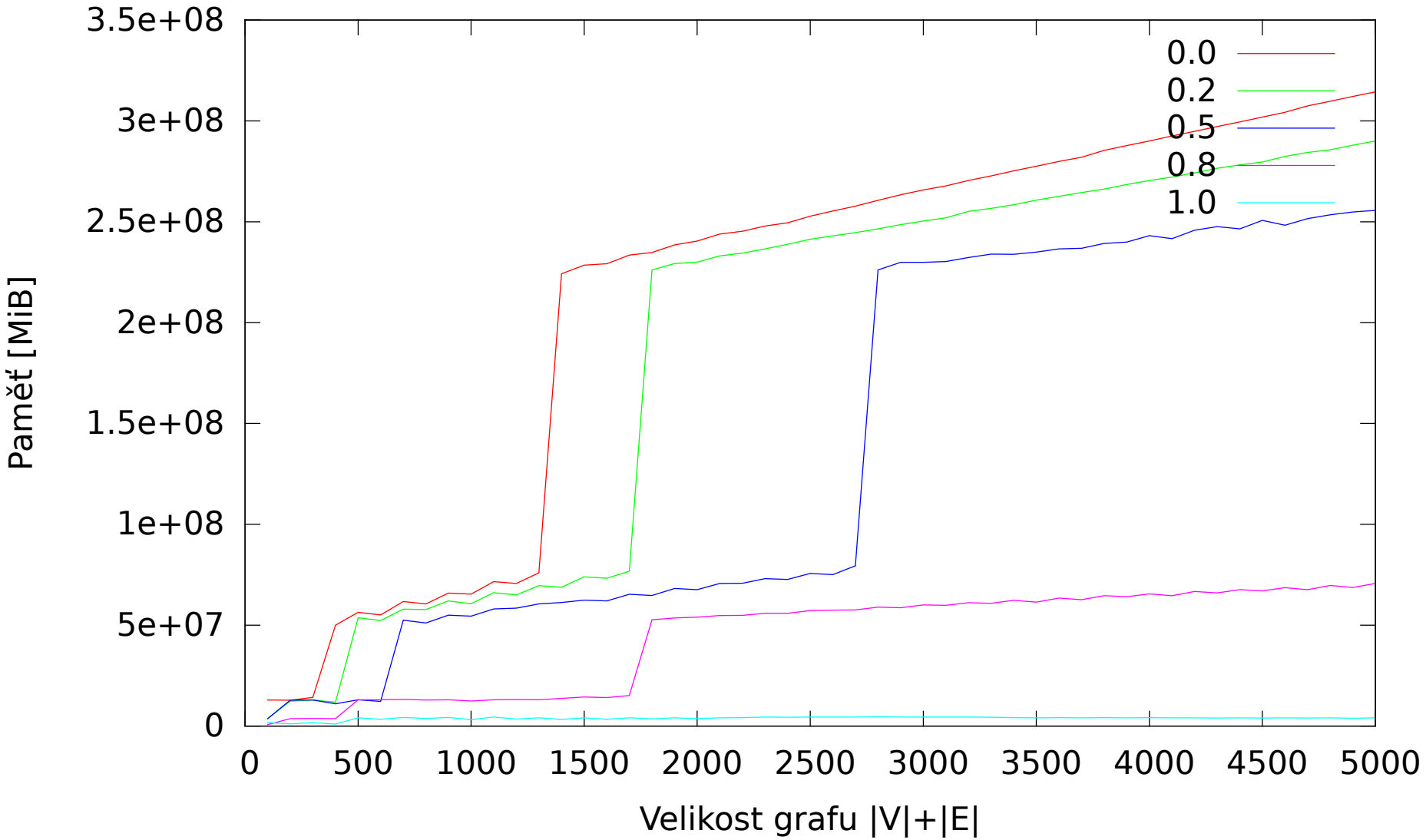


Table of Contents

Gabowův algoritmus



Tarjanův algoritmus

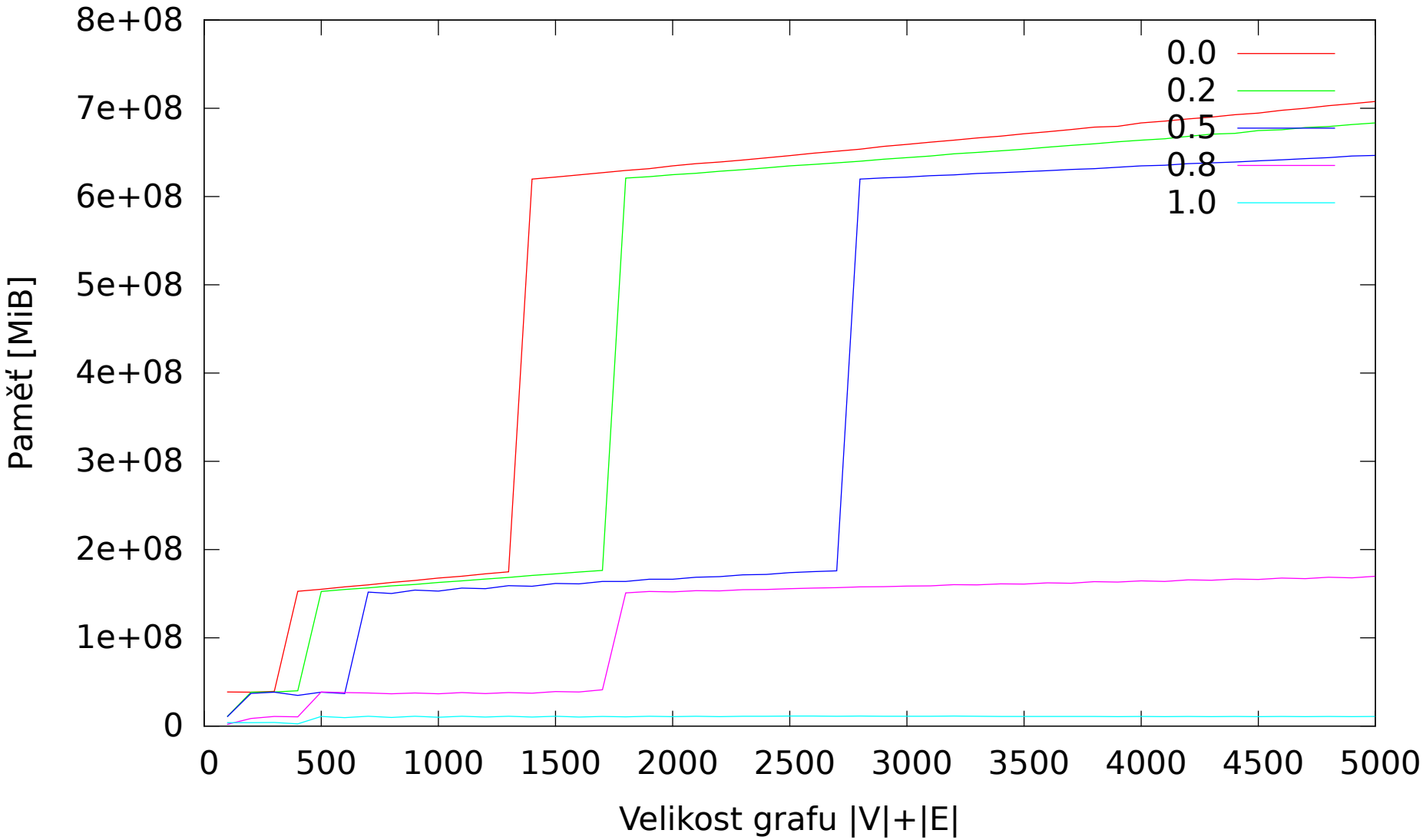
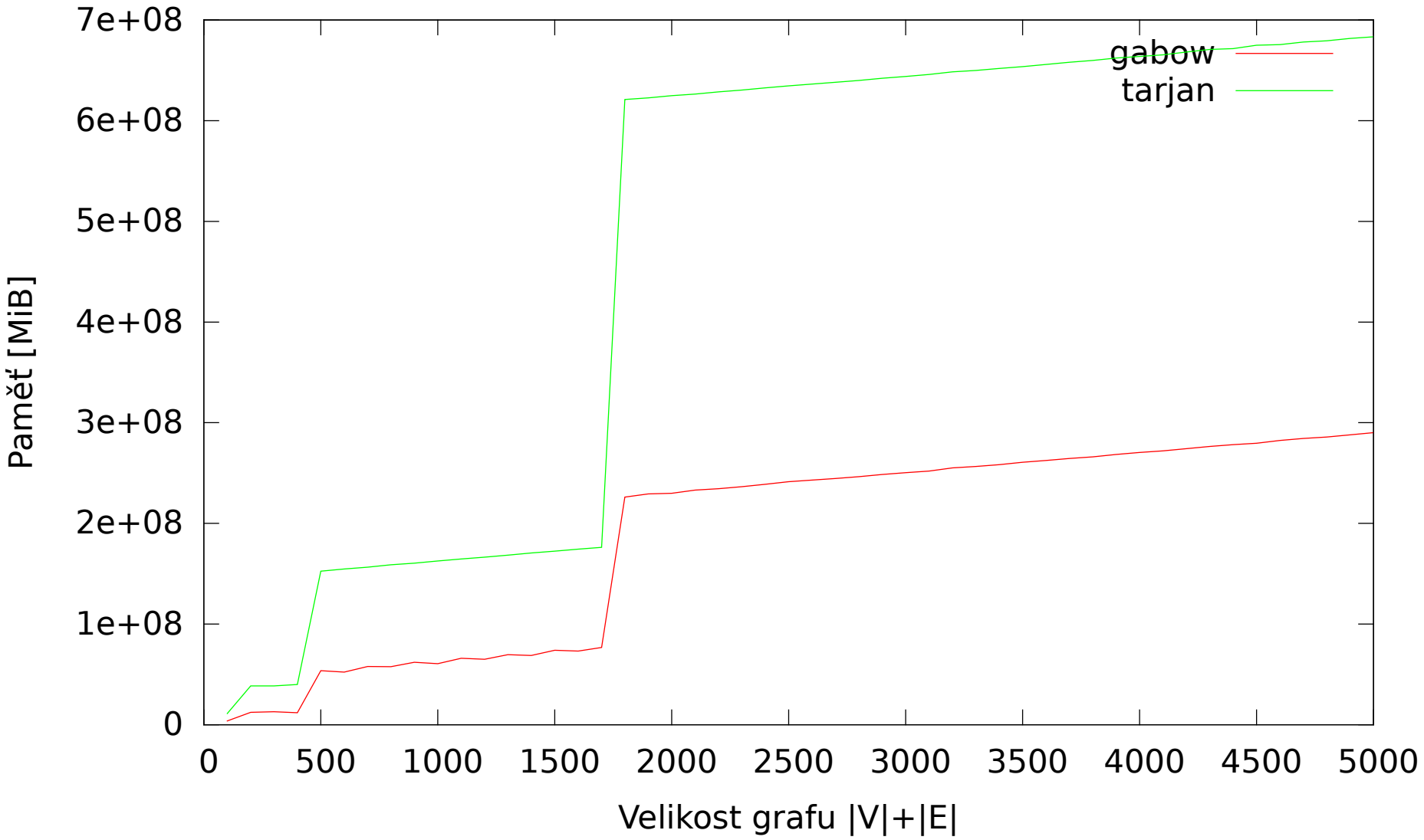


Table of Contents

Srovnání – řídký graf



Srovnání – hustý graf