

# Ataskaita

GitHub nuoroda: <https://github.com/Juliakas/concurrent-3>

## Eksperimentų eiga

Algoritmas pertvarkytas – prieš tai buvo lygiagretinama per *evaluateSolution* funkcijos dalį, kur kiekvienos iteracijos metu apsiukečiama informacija tarp procesų, ko pasekoje atsirado laiko praradimas dėl pranešimų laukimo. Dabar perskaičius failą, visas uždavinys padalinamas į lygias dalis pagal visų įmanomų kombinacijų skaičių (pasirinkti  $r$  kombinacijų naujiems objektams iš  $n$  potencialių vietų). Algoritmas kiekvienam procesui parenka skaičių  $k$ , kuris reiškia nuo kurios  $k$ -osios kombinacijos pradėti geriausio sprendinio paiešką. Sprendinio paieška sustos, kai bus pasiekta sekančio proceso  $k$  reikšmė. Kai visi procesai baigia paiešką, pirmas procesas priima kitų procesų variantus ir atsirenka sprendinį pagal tai, kuris iš jų didžiausias.

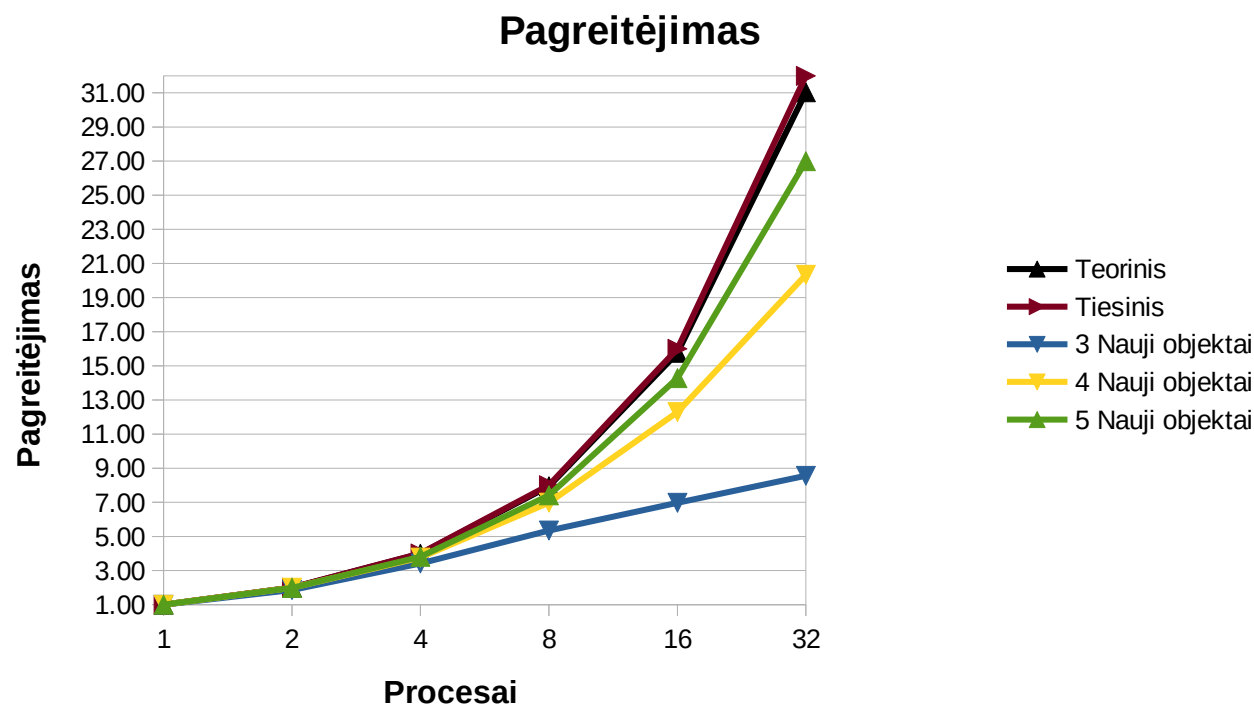
Eksperimentai atlikti cluster sistemoje – parašyti scenarijaus script'ai atitinkamai kiekvienam procesų skaičiui testuoti (16 procesorių scenarijaus [pavyzdys](#)). Script'ai kviečiami pavialu *sbatch run\_\*.sh kartai nauji\_objektai*. Pavyzdžiui *sbatch run\_16.sh 5 3*. Po kurio laiko atsilaivsina eilė ir įvykdoma programa, tada rezultatai matomi out.log faile (arba slurm sugeneruotame faile).

## Algoritmo laiko išmatavimai (sekundėmis):

Laikas													
		Procesai											
		1		2		4		8		16		32	
Nauji objektai	3	6.73	6.60	3.58	3.55	2.09	1.93	1.26	1.24	0.97	0.95	0.76	0.77
		6.41		3.52		1.85		1.22		0.9		0.78	
		6.45		3.54		1.85		1.24		0.99		0.77	
		6.48		3.52		2.01		1.25		0.93		0.78	
		6.95		3.57		1.87		1.21				0.77	
	4	38.03	37.51	19	18.82	10.27	9.98	5.42	5.37	3.05	3.05	1.93	1.84
		38.48		18.92		9.85		5.38		3.03		1.84	
		36.44		18.76		9.9		5.34		3.1		1.82	
		37.56		18.75		9.87		5.39		3.06		1.8	
		37.02		18.68		9.99		5.34		3.02		1.83	
	5	165.57	165.87	83.95	83.56	43.76	43.58	22.22	22.39	11.7	11.62	6.17	6.14
		166.75		83.59		43.75		22.26		11.74		6.1	
		165.71		83.36		43.59		22.09		11.45		6.14	
		165.55		83.74		43.54		22.29		11.57		6.18	
		165.76		83.14		43.27		23.08		11.62		6.13	

# Algoritmo pagreitējimas

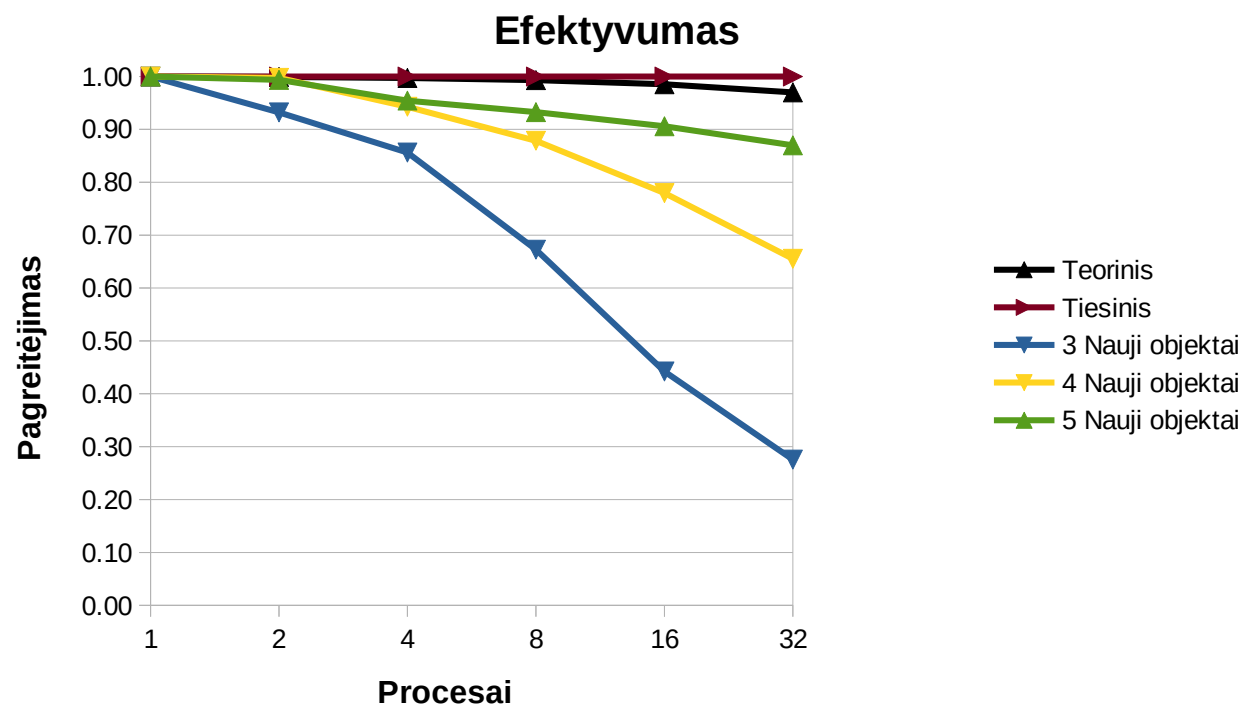
Algoritmo pagreitējimo grafikas:



Lentele pavaizduoti duomenys:

Pagreitējimas							
		Procesai					
		1	2	4	8	16	32
Nauji objektai	3	1.00	1.86	3.41	5.34	6.97	8.55
	4	1.00	1.99	3.76	6.98	12.29	20.34
	5	1.00	1.99	3.81	7.41	14.28	27.00

# Algoritmo efektyvumas



Lentelė pavaizduoti duomenys:

		Efektyvumas					
		Procesai					
		1	2	4	8	16	32
Nauji objektai	3	1.00	0.93	0.86	0.67	0.44	0.28
	4	1.00	1.00	0.94	0.88	0.78	0.66
	5	1.00	0.99	0.95	0.93	0.91	0.87

## **Išvados**

Labiau algoritmas greitėja, kai reikia atlikti skaičiavimus didesniam kiekiui naujų objektų kombinacijai. Kai yra daugiau kombinacijų, sparčiai didėja pačių iteracijų skaičius ir tuo pačiu lygiagretinimo efektyvumas.