Zastosowanie Optymalizacji Ekstremalnej do równoważenia obciążenia procesorów w systemach rozproszonych

dr inż. Eryk Laskowski

Instytut Podstaw Informatyki PAN

21/03/2016

Plan prezentacji

1. Wprowadzenie
2. Omówienie metody EO
3. EO ze Sterowaną Zmianą Stanu (EO–GS)
4. Opis problemu
5. Dynamiczne równoważenie zadań alg. EO-GS
6. Równoległa metoda EO–GS
7. Wyniki badań eksperymentalnych

Współpraca

- Badania prowadzone w Zespole Architektury Komputerowej
  - **prof. dr hab. inż Marek Tudruj**
- Współpraca międzynarodowa:
  - Ivanoe De Falco, Ernesto Tarantino, Umberto Scaffuri – Institute of High Performance Computing and Networking, CNR, Neapol, Włochy
  - Richard Olejnik – Computer Science Laboratory, University of Science and Technology of Lille, Francja

Wprowadzenie

- Extremal Optimization (EO)
  - **ewolucyjna metoda optymalizacji**, autorzy: Boettcher i Percus, 1999 r.,
  - inspiracją jest model tzw. „self-organized criticality”, Bak-Sneppen, 1987 r.
- Zalety w stosunku do innych metod (GA, PSO, DE oraz SA, TS itd.)
  - niska złożoność czasowa,
  - małe wymagania pamięciowe.
- Nie stosowana do równoważenia zadań.

Ogólne zasady EO

- Rodzaj przeszukiwania stochastycznego, z **pojedynczym osobnikiem** reprezentującym rozwiązanie (chromosomem).
- Zmiana stanu: aktualizacja **najgorszych składowych** rozwiązania
  - zastępowane przez wartości losowe (brak jawnego ulepszania, model Bak-Sneppen).
- Dwie funkcje dopasowania
  - **lokalna** – ocena składowych rozwiązania,
  - **globalna** – ocena całego osobnika.

τ–EO

- Metoda unikania lokalnych optimów
  - składowe są sortowane według rosnącej wartości lokalnej funkcji dopasowania,
  - prawdopodobieństwo wylosowania składowej k: $p_k \sim k^{-\tau}$,
  - składowa jest losowana a następnie zmieniania losowo,
  - osobnik jest zmieniany na $S' \in Neigh(S)$ bezwarunkowo.
- Parametry algorytmu: liczba iteracji N_{iter} i τ.

Schemat algorytmu EO

- initialize configuration S at will
- S_{best} <- S
- WHILE total number of iterations N_{iter} not reached
  - evaluate φ_i for each variable s_i of the current solution S
  - rank the variables s_i based on their local fitness φ_i
  - choose the rank k according to $k^{-\tau}$ so that the variable s_j with $j = \pi(k)$ is selected
  - choose $S' \in Neigh(S, s_j)$ such that s_j must change
  - accept S <- S' unconditionally
  - IF $\Phi(S) < \Phi(S_{best})$
  - S_{best} <- S
  - ENDIF
- ENDWHILE
- RETURN S_{best} and $\Phi(S_{best})$

EO ze Sterowaną Zmianą Stanu

- Ulepszona wersja algorytmu EO (ang. EO with Guided State Changes)
  - **motywacja:** trudności z szybkim znalezieniem dobrego rozwiązania dla dużej liczby jednostek obliczeniowych w systemie,
  - klasyczny EO stosuje w **pełni losową selekcję** nowego rozwiązania w każdym kroku metody,
  - propozycja: zamiana wyboru losowego na wybór stochastyczny z rozkładem prawdopodobieństwa określonym **przez funkcję wykorzystującą** wiedzę dziedzinową.

EO with Guided State Changes

- initialize configuration S at will, S_{best} <- S
- WHILE total number of iterations N_{iter} not reached
  - evaluate φ_i for each variable s_i of the current solution S
  - assign ranks k to the variables s_i based on their local fitness φ_i
  - choose the rank k stochastically according to $k^{-\tau}$ so that the variable s_j with $j = \pi(k)$ is selected for improvement
  - evaluate some problem knowledge function ω_s on neighbours $S_v \in Neigh(S,s_j)$, generated by changes of s_j in the current solution S
  - assign ranks g to the neighbours $S_v \in\in Neigh(S,s_j)$, defined with the use of ω_s
  - choose $S' \in Neigh(S,s_j)$ according to an exponential distribution $Exp(g, \lambda) = \lambda e^{-\lambda g}$
  - accept S <- S' unconditionally
  - IF $\Phi(S) < \Phi(S_{best})$
  - S_{best} <- S
  - ENDIF
- ENDWHILE
- RETURN S_{best} and $\Phi(S_{best})$

Problem równoważenia obciążenia

- **klaster *N*** jednostek obliczeniowych /wielordzen./ połączonych siecią komunikacyjną /przes. komun./,
- **zbiór zadań *T*** /wątków/
- **problem:**
  - przypisz zadania t_k, $k \in 1...|T|$ do węzłów n, $n \in [0, N - 1]$ w taki sposób, by zminimalizować całkowity czas działania programu,
- **równoważenia obciążenia** polega na wykonywaniu kroków **detekcji i korekcji**,
- zrównoważenie obciążenia ulepsza się poprzez **migrację zadań** między jednostkami obliczeniowymi.

Wykrywanie niezrównoważenia

- Opiera się na badaniu **różnicy dostępności mocy procesora** w poszczególnych jednostkach obliczeniowych
  $LI = max\ (TimeCPU(n)) - min\ (TimeCPU(n)) \geq \alpha$
  - *TimeCPU(n):* indeks aktualnej dostępności procesora, tj. procent mocy obliczeniowej CPU dostępnej dla wątków aplikacji na węźle *n*, szacowany na podst. periodycznej obserwacji wykonywanej przez /wątki/ agentów systemowych,
  - α – współczynnik określony eksperymentalnie (stosowaliśmy wartość z zakresu 25%-75%).

Model aplikacji

- Aplikacja: **zbiór zadań** ***T***
- Zadanie t_k, $k \in 1...|T|$ może być procesem lub wątkiem
- Dwie metryki
  - **com(t_1, t_2)** – rozmiar komunikacji między zadaniami
  - **wp(*t*)** – miara obliczeń, realizowanych przez zadanie
  - dostarczane przez twórcę aplikacji, wystarczy, że są szacunkami.

Lokalna funkcja dopasowania

- **Lokalna funkcja dopasowania:**

$$\varphi(S) = \gamma * load(\mu_t) + (1 - \gamma) * rank(t)$$

  - $load(\mu_t)$ – wskazuje jak bardzo obciążenie węzła μ_t przekracza średnie obciążenie węzłów
  - $rank(t)$ – wskazuje czy zadanie t jest „dobrym” kandydatem do migracji (średni rozmiar, mało komunikacji z zadaniami aktualnego węzła).

Globalna funkcja dopasowania

- **Globalna funkcja dopasowania:**

$$\Phi(S) = attrExtTotal\ (S) * \Delta_1 + migration\ (S) * \Delta_2 + imbalance\ (S) * [1 - (\Delta_1 + \Delta_2)]$$

  - *attrExtTotal (S)* – określa wpływ komunikacji zewnętrznej na jakość rozwiązania
  - *migration (S)* – metryka kosztów migracji
  - *imbalance (S)* – miara niezrównoważenia dla rozwiązania S.

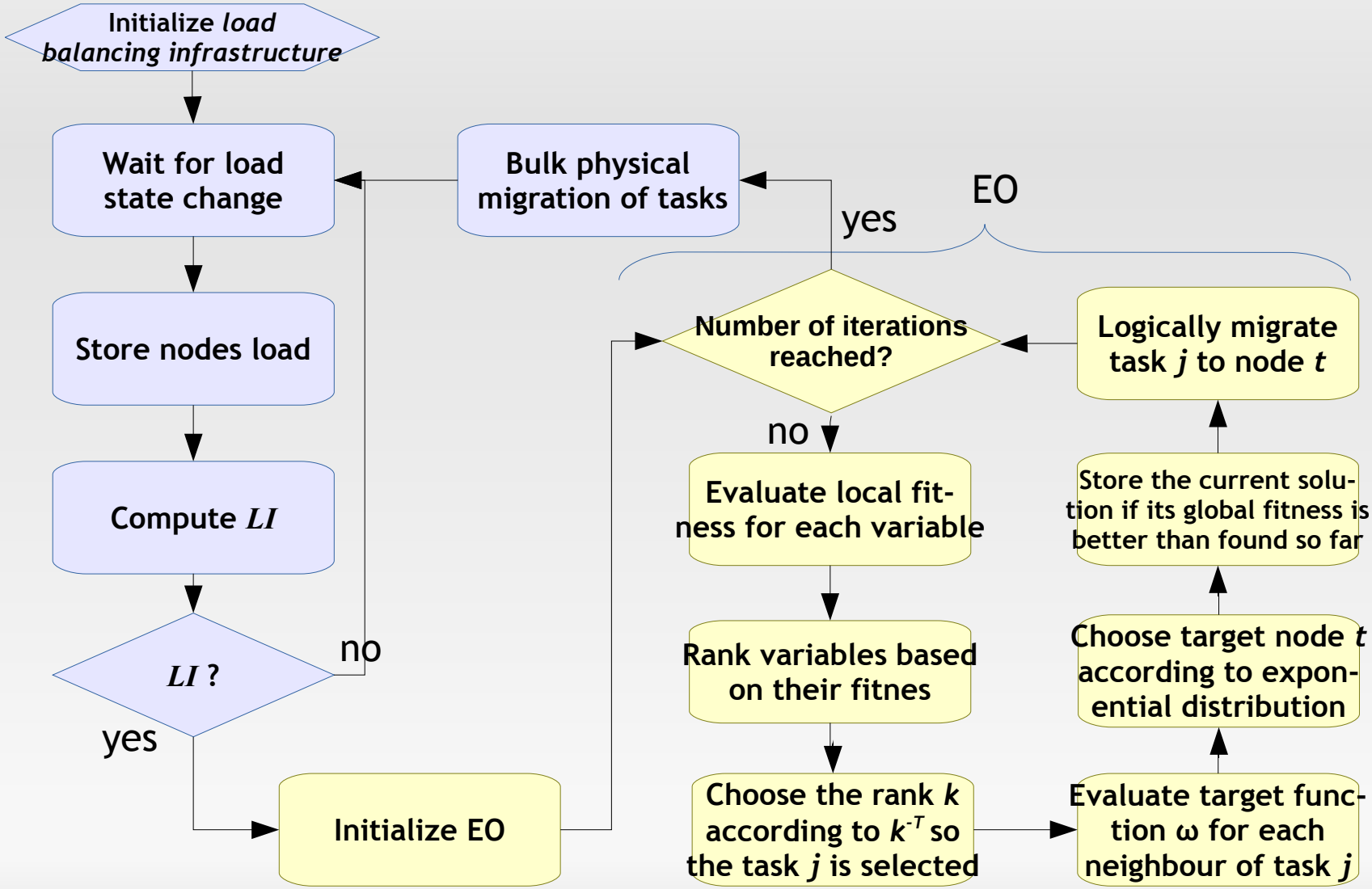
Sterowana zmiana stanu

- W klasycznym EO nowe rozwiązanie losowane jest z rozkładem równomiernym.
- W alg. EO-GS (EO with **guided state changes**) wybór nowego rozwiązania również odbywa się losowo, lecz rozkład zależy od pewnej funkcji dziedzinowej, w celu preferowania określonych sąsiadów.
- Po posortowaniu węzłów docelowych, jeden z nich jest wybierany losowo jako nowe położenie zadania – zgodnie z rozkładem wykładniczym $Exp(g, \lambda) = \lambda e^{-\lambda g}$

Sterowana zmiana stanu

- Przy każdej aktualizacji komponentu rozwiązania:
  - węzły systemu sortujemy zgodnie z pewnym rankingiem GS,
  - implementacja: funkcja porównania $\omega(n_1, n_2)$, $n_1, n_2 \in N$,
- Co preferujemy:
  - węzły **nieobciążone** (lub obciążone mniej niż średnia),
  - węzły, z którymi następuje **intensywna komunikacja** aktualnego zadania.

Schemat równoważenia EO–GS



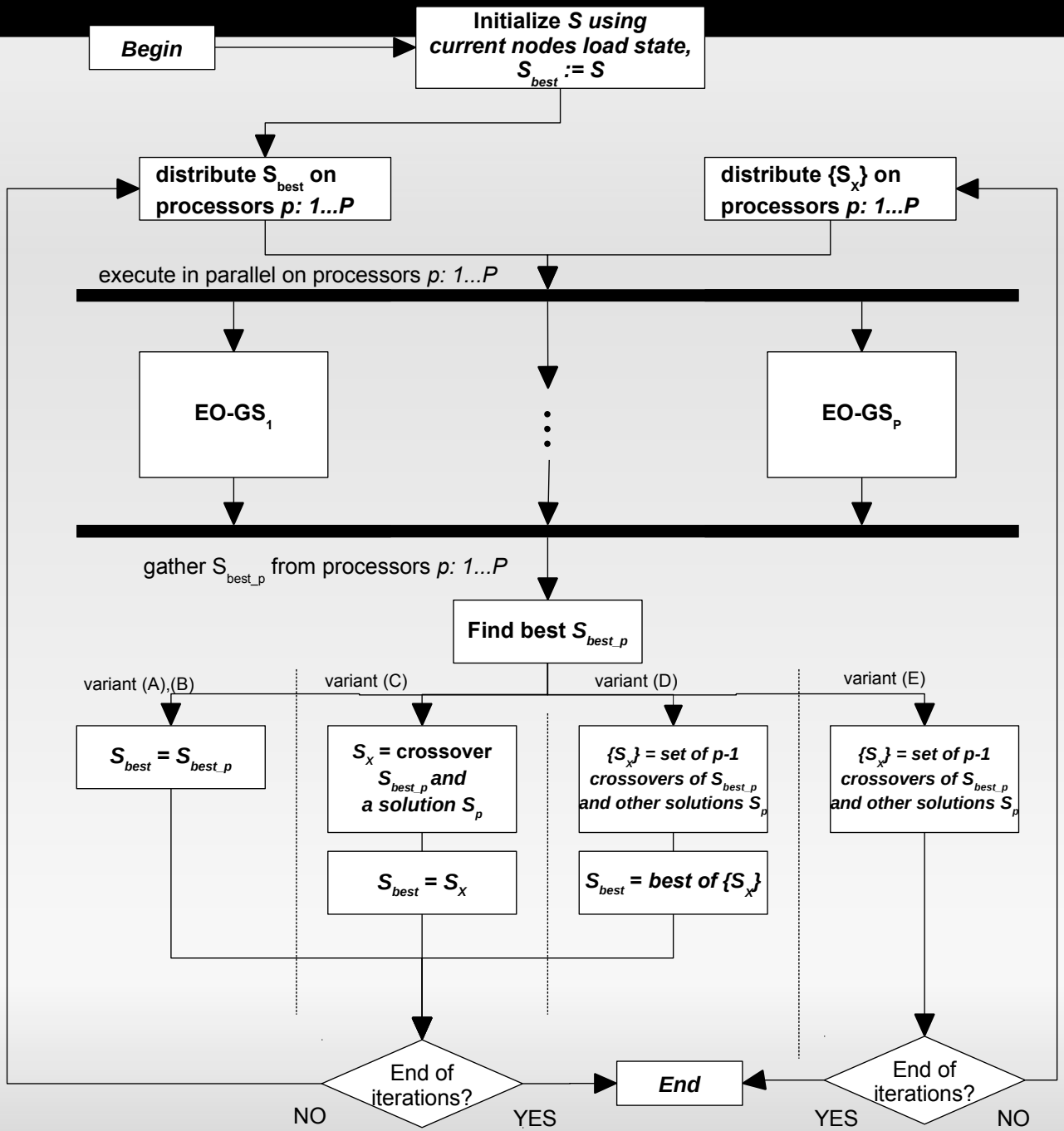
Wersja równoległa algorytmu EO

- Oparta na modelu populacyjnym EO
  - jednocześnie istnieje wiele rozwiązań,
  - równoległe ulepszania rozwiązań w całej populacji.
- Wymiana rozwiązań między populacjami
  - wybór „najlepszego” rozwiązania spośród wielu rozwiązań, ulepszanych równolegle w obrębie populacji EO,
  - następuje cyklicznie, po pewnym zadanym interwale iteracji.

Wybór rozwiązania do wymiany

- Wybór rozwiązania do wymiany wykorzystuje S_{best} z bieżącej iteracji.
- Warianty wyboru rozwiązania startowego:
  - (A) najlepsze znalezione rozwiązanie
  - (B) „średnio” najlepsze
  - (C) wynik krzyżowania najlepszego rozwiązania oraz innego, wybranego losowo
  - (D) wybierz najlepsze z krzyżowania S_{best} z pozostałymi rozwiązaniami
  - (E) rozprosz wynik krzyżowania S_{best} z pozostałymi rozwiązaniami.

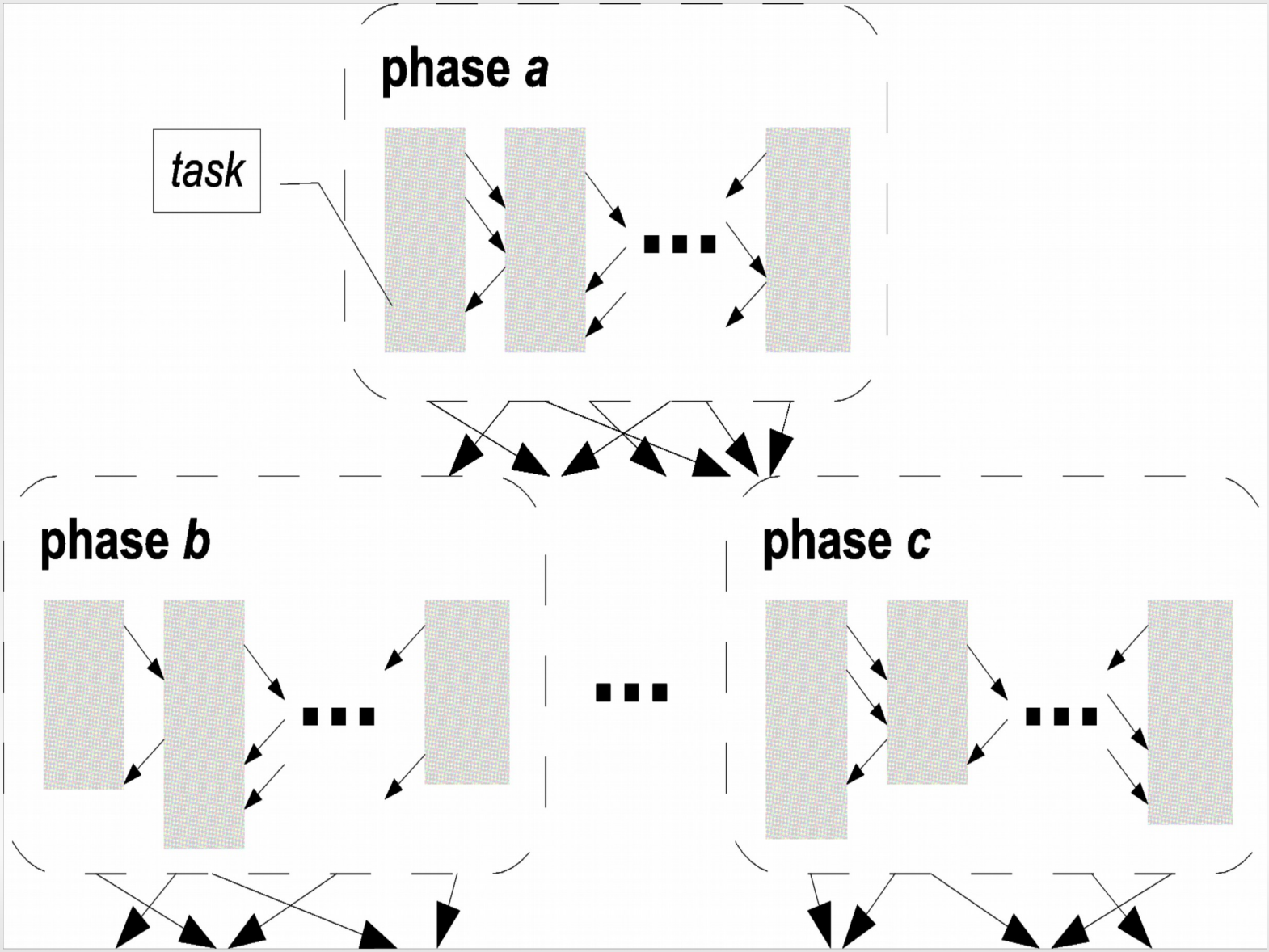
Schemat równoległego alg. EO



Badania eksperymentalne

- Zastosowaliśmy podejście **symulacyjne**
  - symulujemy system poddawany równoważeniu.
- Charakterystyka systemu wykonawczego
  - klaster wielordzeniowych jednostek obliczeniowych, z komunikacją opartą na przesyłaniu komunikatów
  - przykład systemu: zespół stacji roboczych, z procesorami wielordzeniowymi, Linux, biblioteka komunikacyjna MPI.
- Symulator oparty na **modelu DEVS** (symulacja sterowana zdarzeniami na dyskretnej osi czasu).
- Syntetyczne aplikacje testowe.

Struktura aplikacji testowych



Badania eksperymentalne cd.

- Dwa zbiory zadań
  - 8 zadań o rozmiarze 128-576 zadań, nieregularne i regularne,
  - 4 zadania o rozmiarach 1000, 5000, 10000, 20000 zadań,
  - współczynnik komunikacja/obliczenia w zakresie [0.10, 0.20].
- Badania:
  - własności algorytmów,
  - porównawcze.

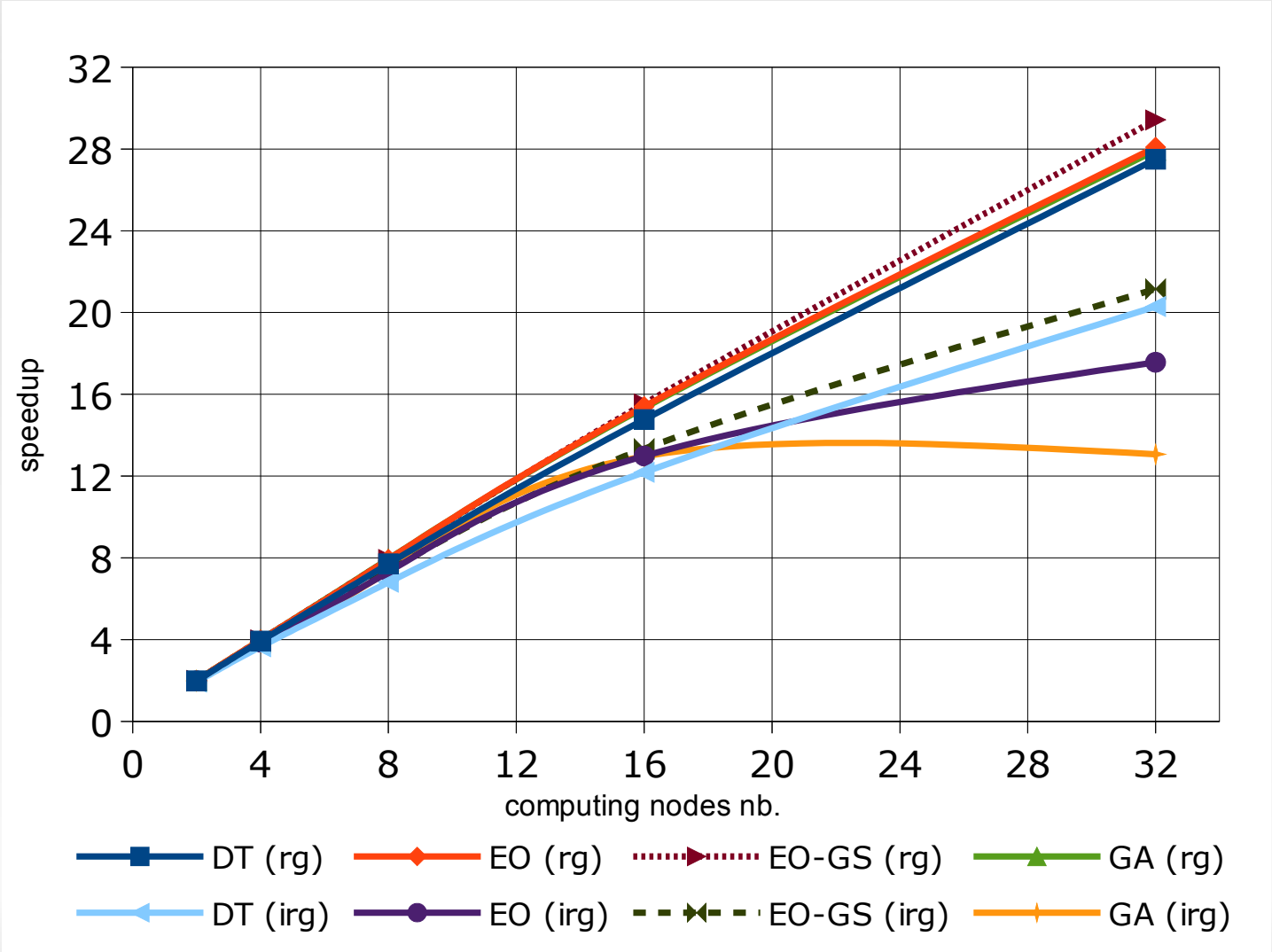
Badania eksperymentalne cd.

- Parametry algorytmów równoważenia obciąż.:
  - „standardowe” $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.75$, $\Delta_1 = 0.13$, $\Delta_2 = 0.17$, $\tau = 1.5$, dla EO–GS $\lambda = 0.5$
  - IF5-IF9 – zestawy wartości β, γ, Δ_1, Δ_2
- Każdy eksperyment był powtarzany 20 razy, z różnymi początkowym konfiguracjami zadań aplikacji testowej.
- Badania szerokiego zakresu liczby iteracji 30-4000, przy czym większość badań wykonano dla $N_{\text{iter}} = 500$.

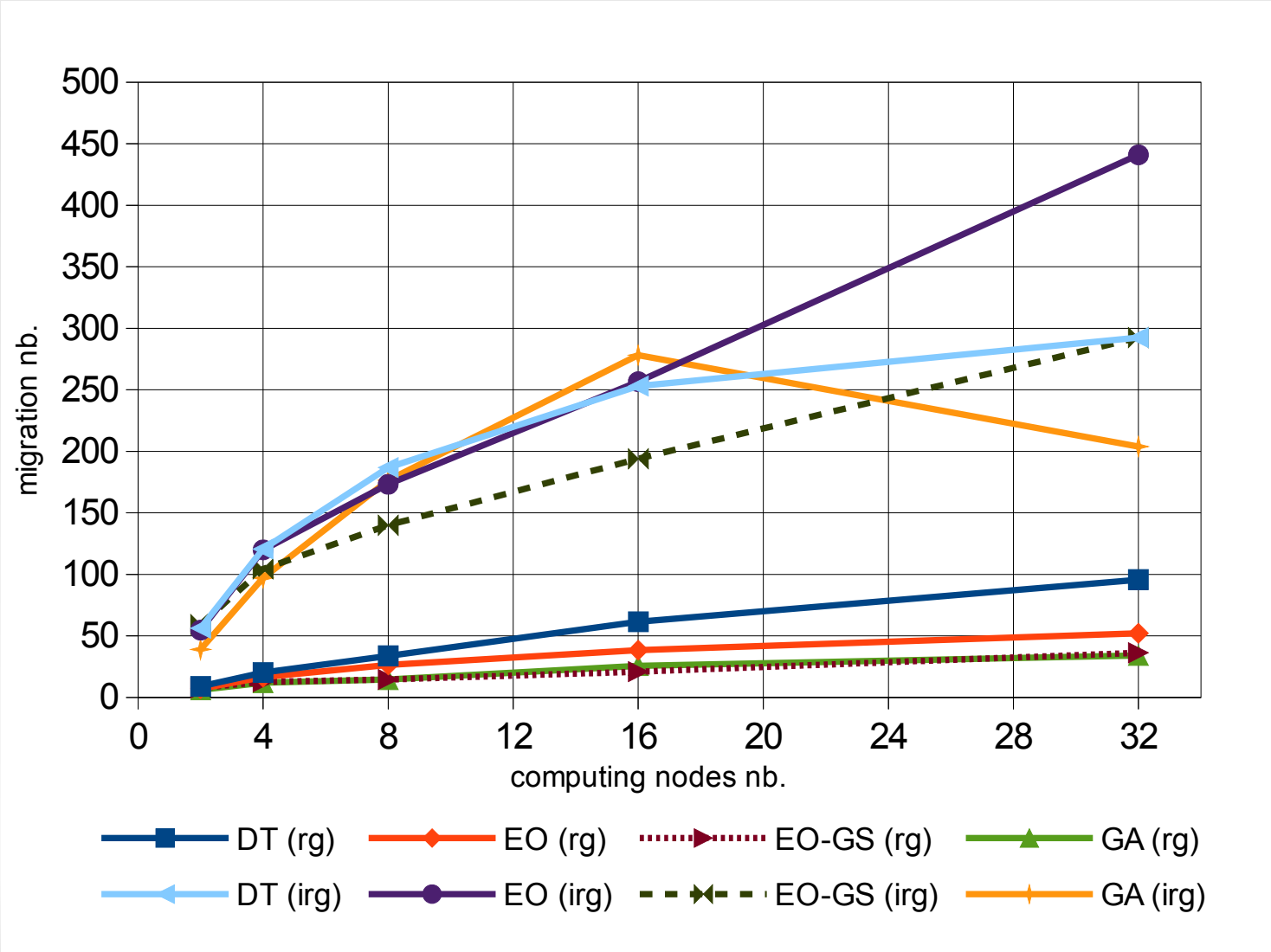
Badania eksperymentalne cd.

- Co badaliśmy:
  - EO, EO-GS
  - PEO-A, PEO-GS-A, PEO-GS-B
  - PEO-GS-C, -D, -E
- Badania porównawcze:
  - GA
  - DT
- Rozmiar systemu wykonawczego:
  - 2-32 /małe aplikacje/, 32-128 /duże aplikacje/

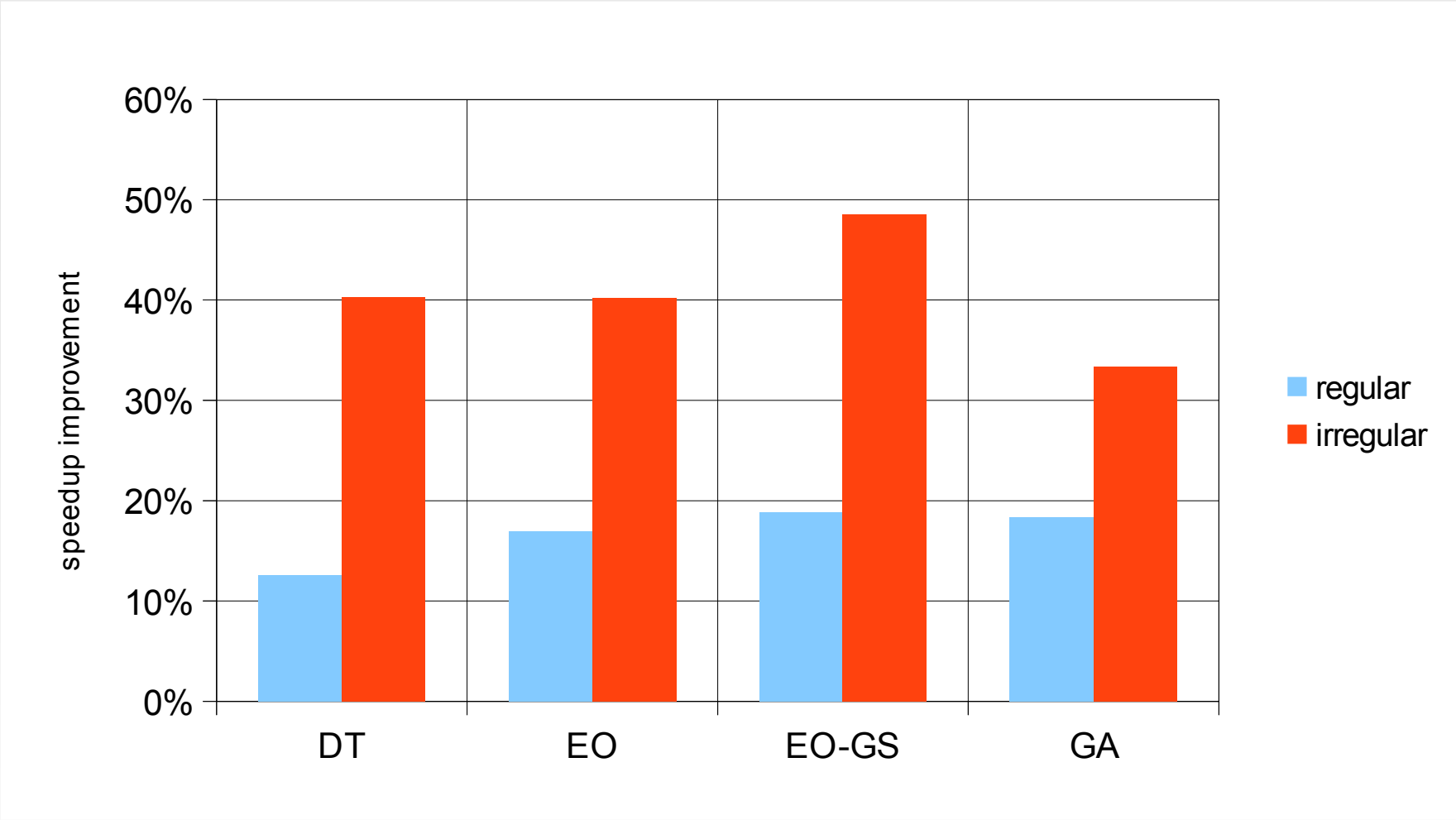
Przyśpieszenie



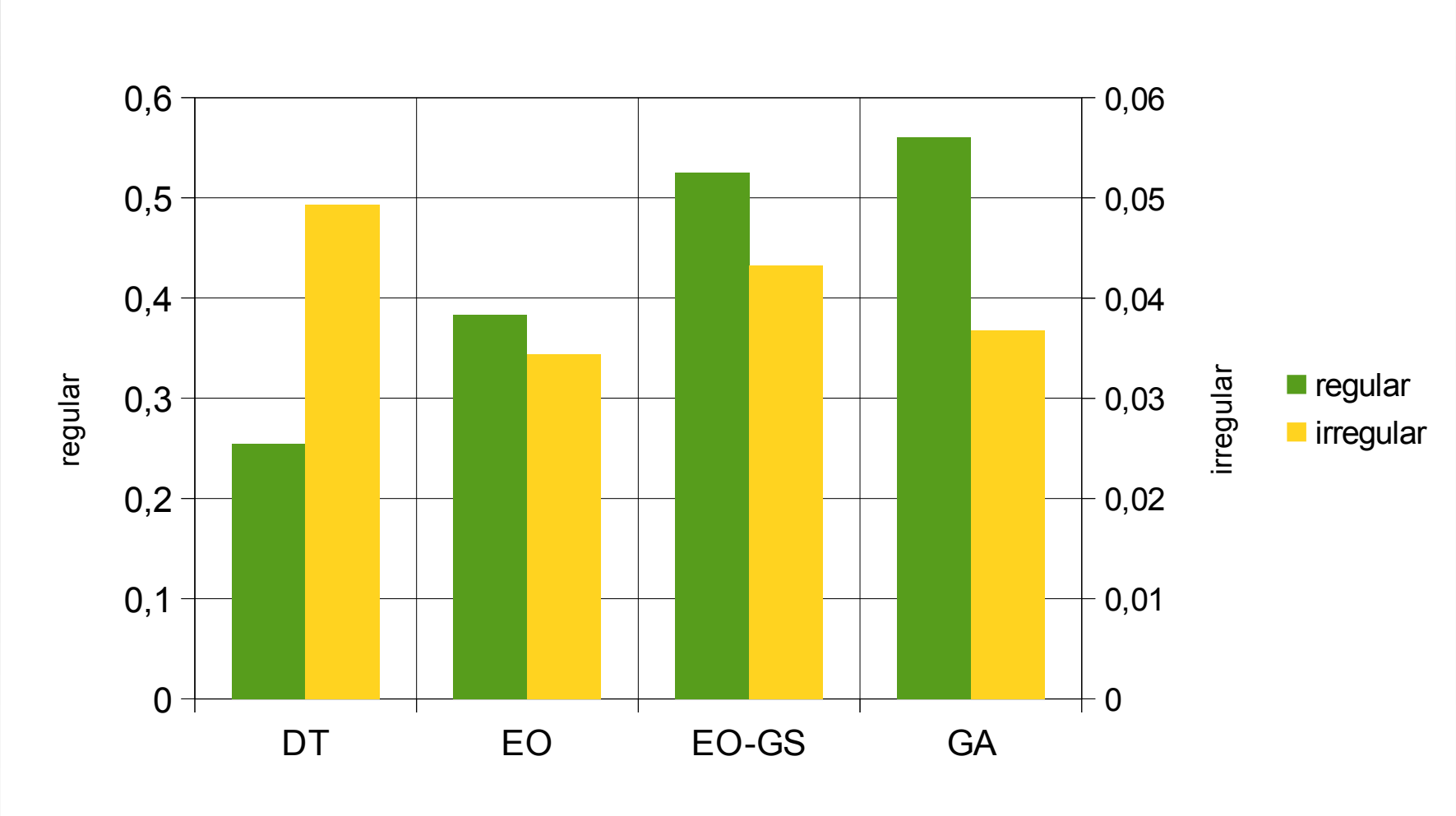
Migracje


500
450
400
350
300
250
200
150
100
50
0
migration nb.
0
4
8
12
16
20
24
28
32
computing nodes nb.
DT (rg)
EO (rg)
EO-GS (rg)
GA (rg)
DT (irg)
EO (irg)
EO-GS (irg)
GA (irg)

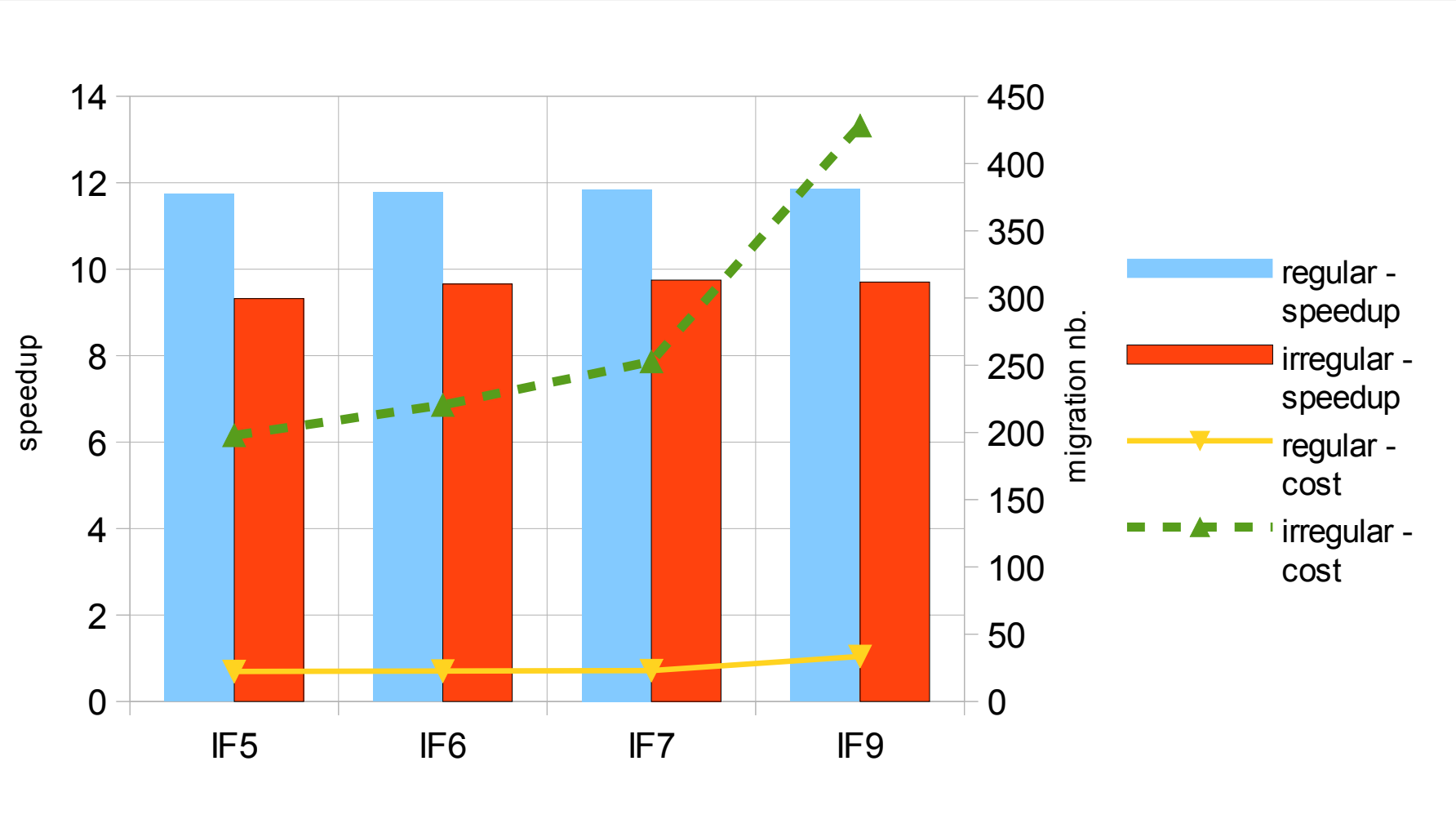
Poprawa przyśpieszenie (średnia)



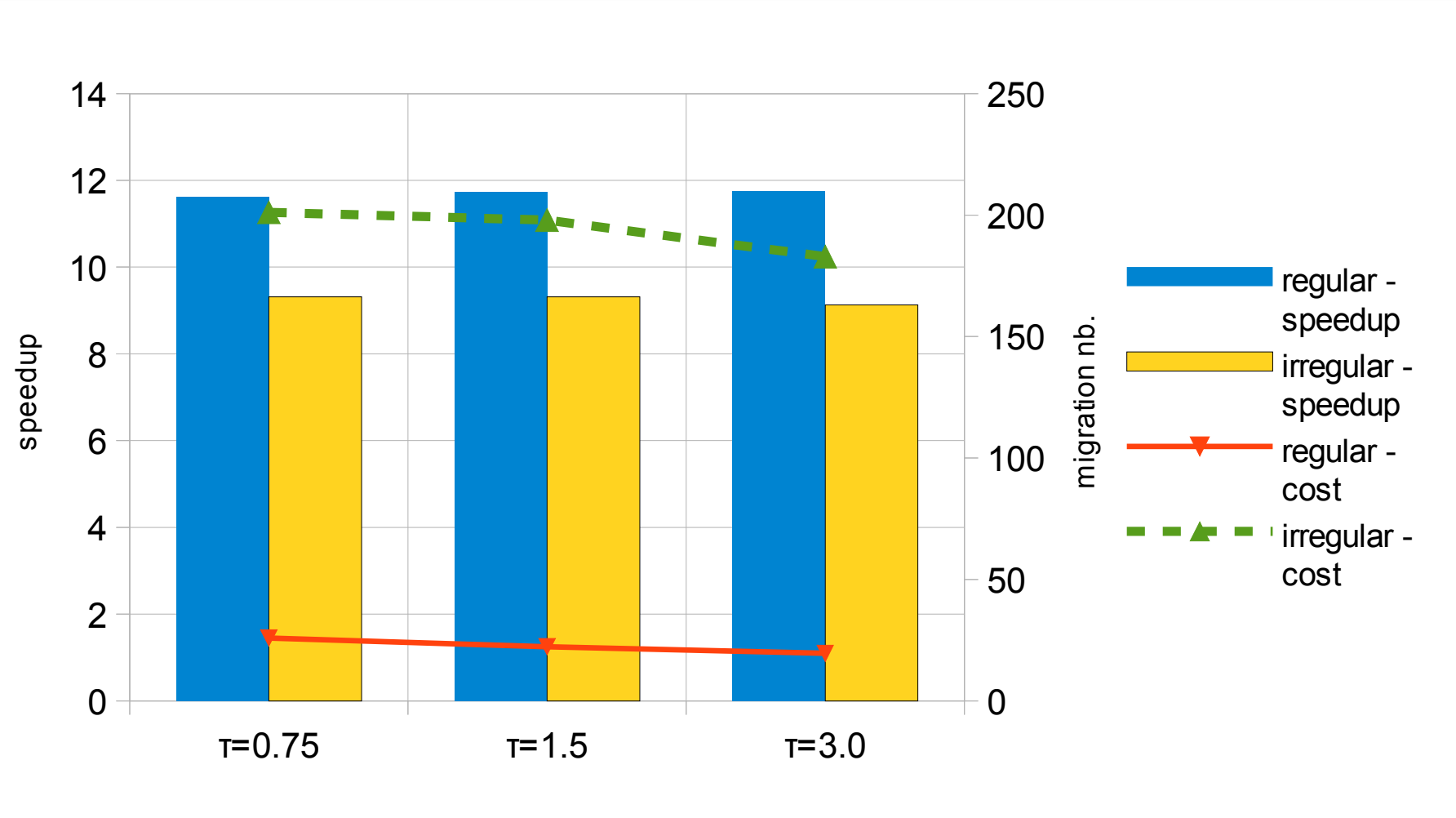
Efektywność migracji (średnia)



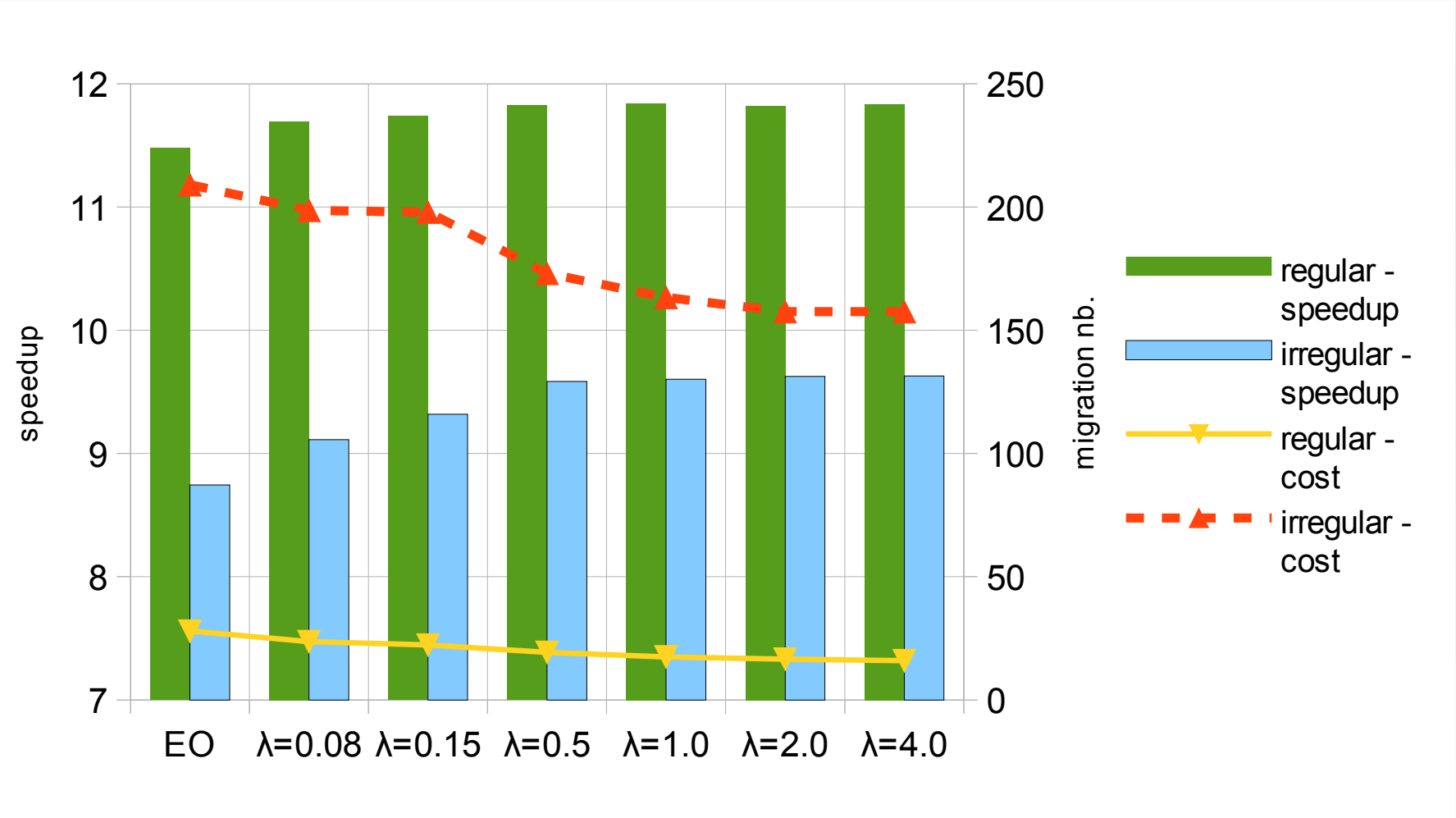
Wpływ wart. parametrów alg. LB



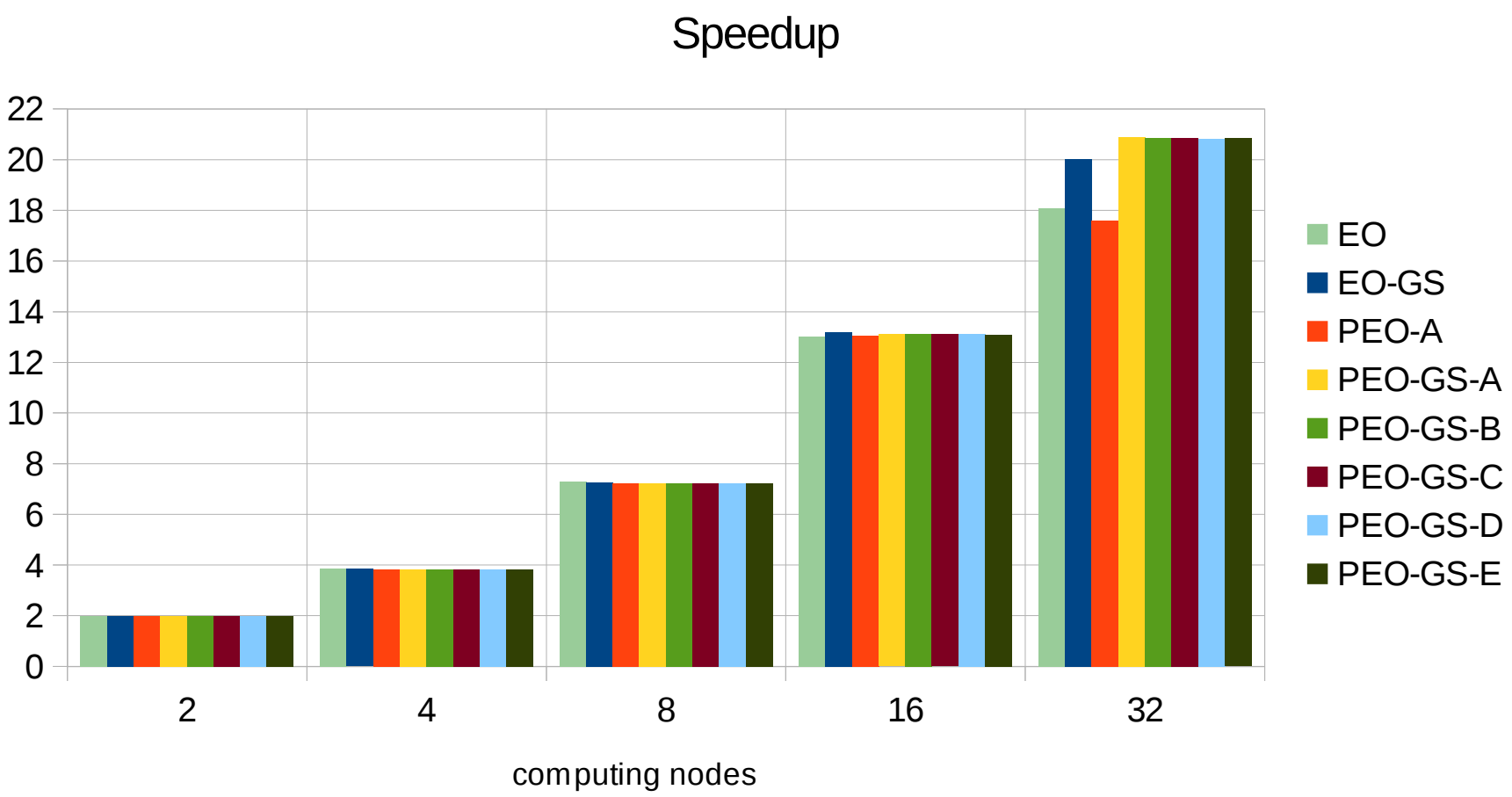
Wpływ wartości tau na wynik



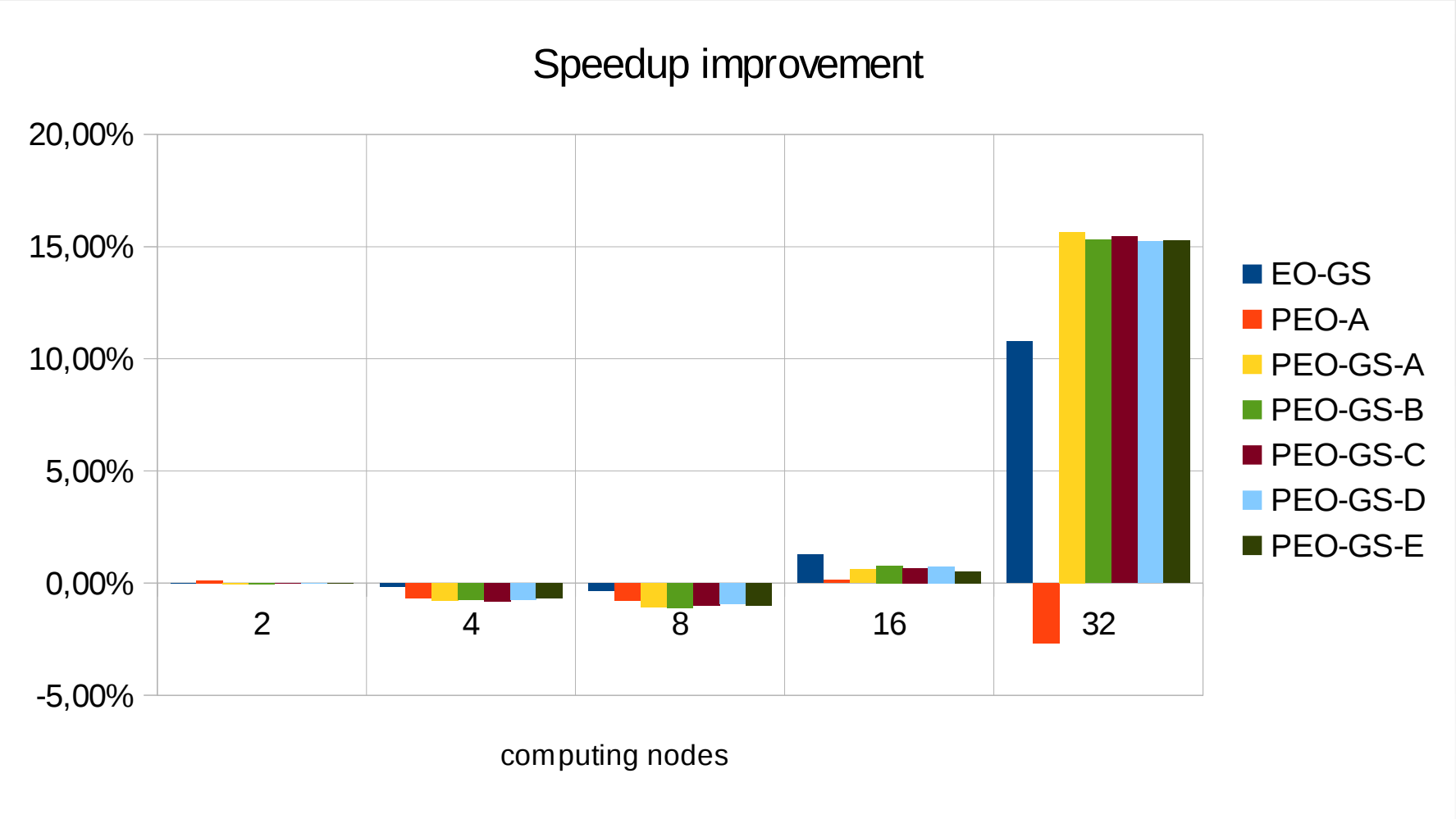
Wpływ wartości lambda na wynik



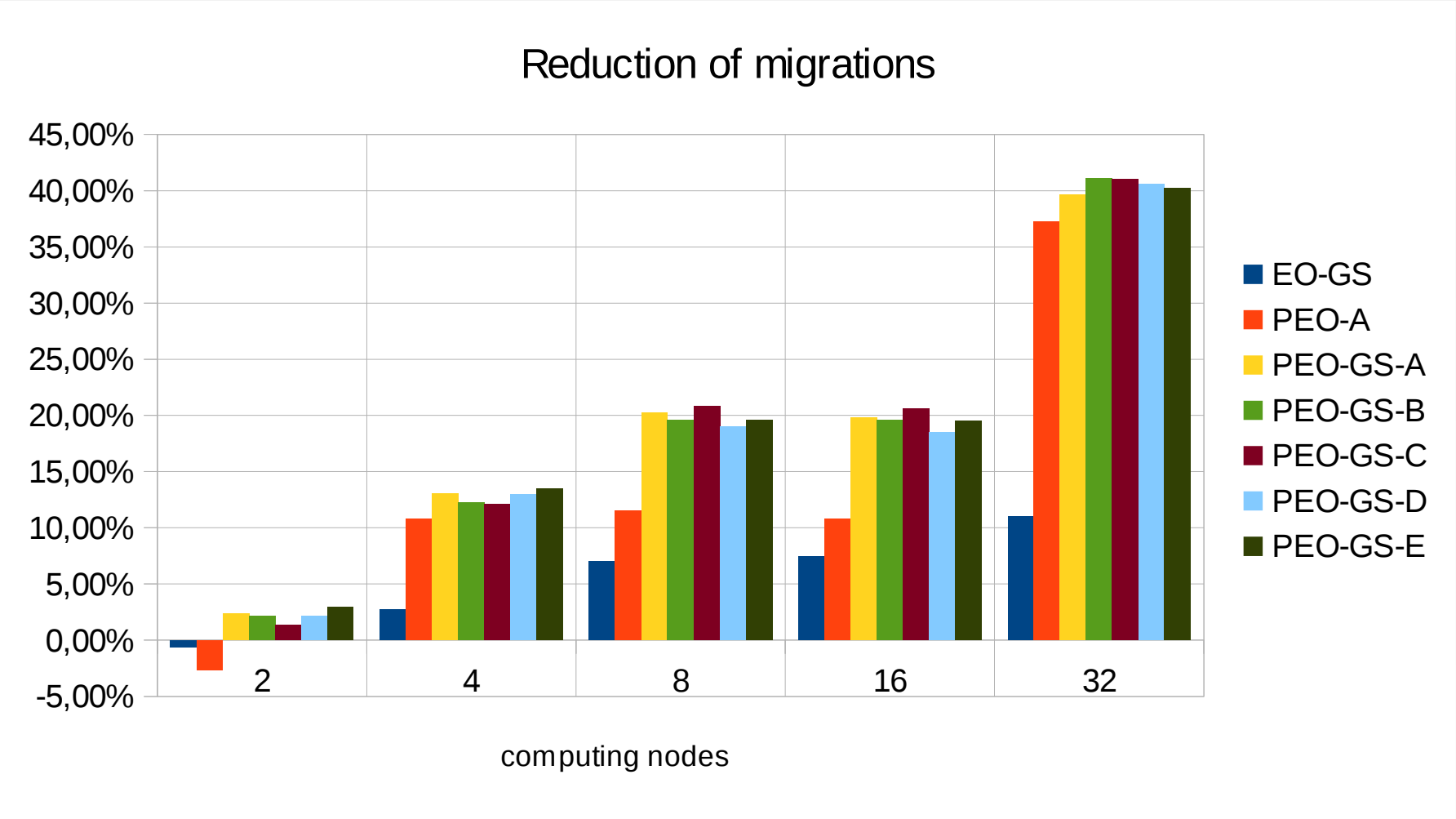
Przyśpieszenie



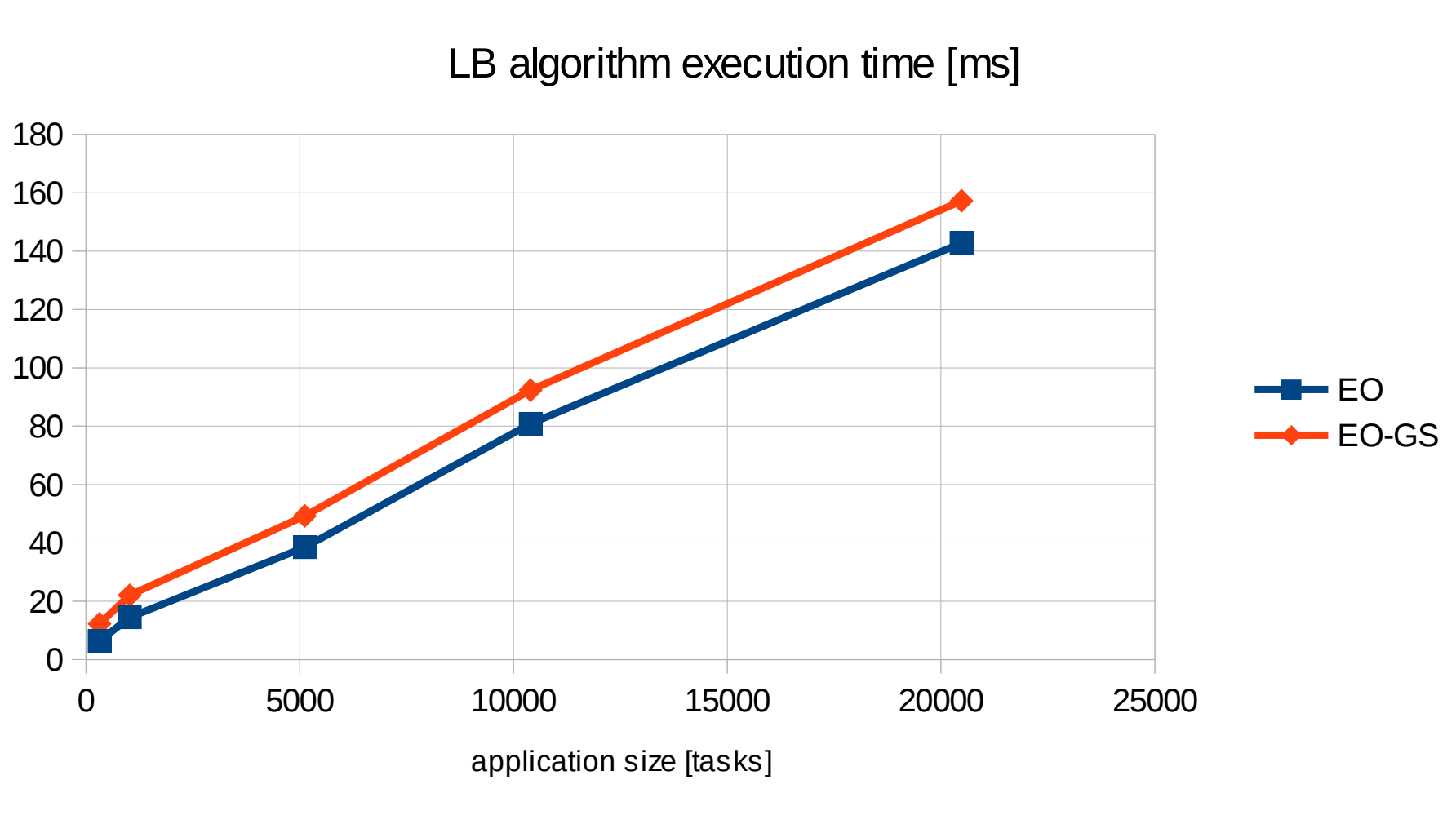
Poprawa przyśpieszenia wzgl. EO



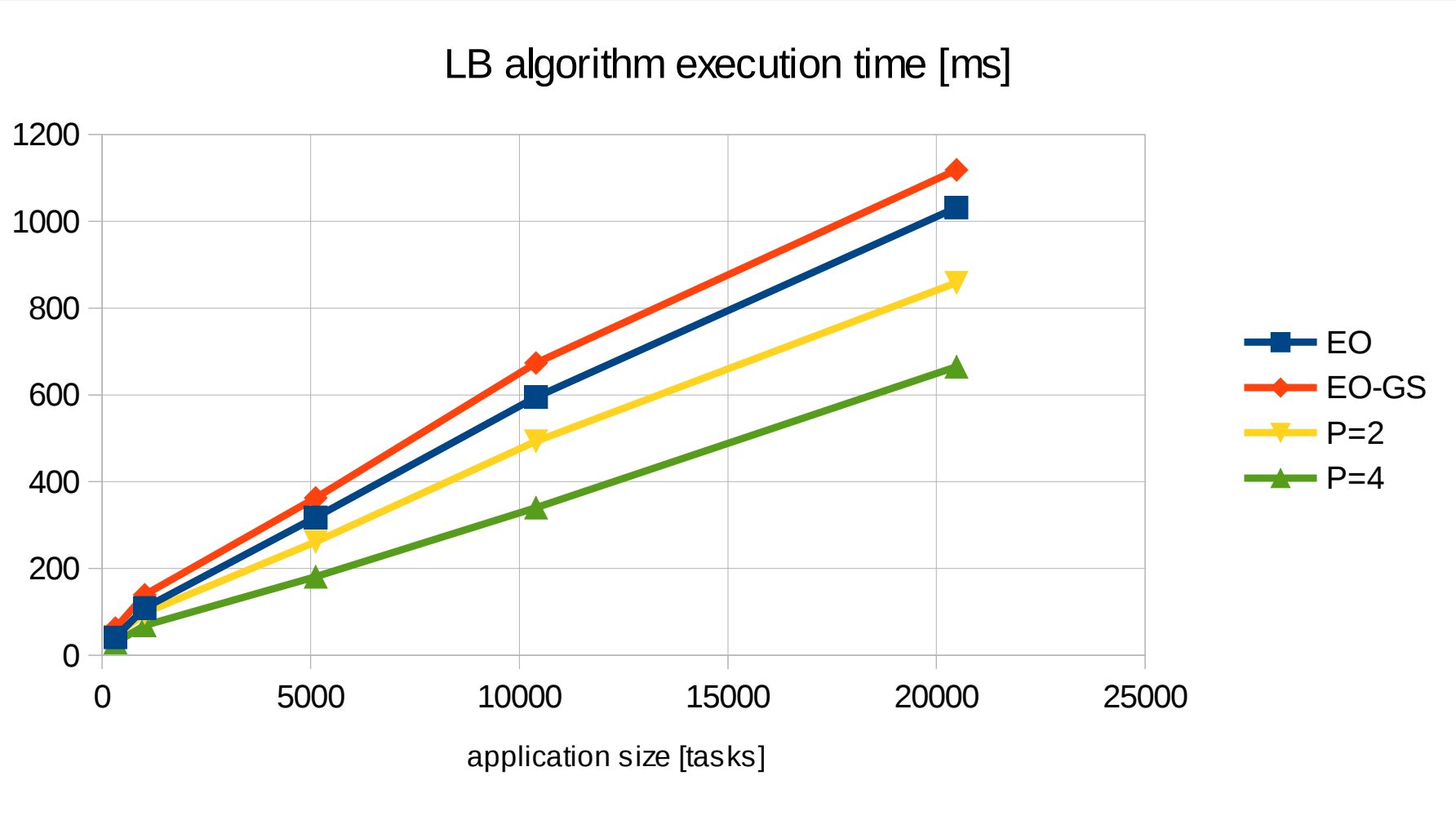
Redukcja migracji wzgl. EO



Średni czas działania N_{iter} = 500



Średni czas działania N_{iter} = 4000



Podsumowanie

- Metoda EO i jej wariant EO-GS doskonale nadaje się do równoważenia zadań w systemach równoległych.
- Równoległe EO i EO-GS pozwalają skrócić czas działania algorytmu, przy jednoczesnym niewielkim polepszeniu wyników.
- Pozwala to na zastosowanie EO w algorytmach dynamicznego równoważenia obciążeń procesorów, gdzie niska złożoność algorytmów ma nadrzędne znaczenie.
- Skalowalność do dziesiątków tysięcy zadań.
- Redukcja liczby migracji zadań.

Dziękuję za uwagę!