Równoważenie obciążenia w środowisku JPPF (ang. Java Parallel Processing Framework) z wykorzystaniem optymalizacji ekstremalnej

dr inż. Eryk Laskowski

Instytut Podstaw Informatyki PAN

25/04/2019

Plan prezentacji

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

Trochę historii

- **Badania prowadzone w Zespole Architektury Komputerowej – doktorat**
  - eksperymenty z alg. ewolucyjnymi – algorytmy genetyczne
- **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

Trochę historii cd.

- **Lata 2007 – 2010: zastosowanie EO do szeregowania zadań**
  - połowiczny sukces
- **2012-2014: budowa środowiska symulacyjnego**
- **Od 2013: zastosowanie EO do równoważenia obciążeń obliczeniowych**
- **Od 2017: podejście wielokryterialne**
- **Jesień 2018 – JPPF**
- **“Po drodze” warianty równolegle oraz kilka wariantów funkcji oceny ww. algorytmów**

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.)**
  - wysoka wydajność obliczeniowa,
  - małe wymagania pamięciowe.
- **Nie stosowana uprzednio 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 $\Phi(S_{best})$**

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 $Exp(g, \lambda) = \lambda e^{-\lambda g}$

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:**

$$\varphi(S) = y \times load(\mu_t) + (1 - y) \times rank(t)$$

  - $load(\mu_t)$ – 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:**

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

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

Implementacja sterowanej zmiany 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.

Optymalizacja wielokryterialna

- **Metoda wyznaczania najlepszego rozwiązania (“optymalnego”) z punktu widzenia kilku** kryteriów
  - ang. *multi-objective/multicriteria/multiattribute optimization*
  - zazw. poszukiwanie ekstremum funkcji
  - kryteria mogą być sprzeczne, może nie istnieć rozwiązanie optymalizujące jednocześnie wszystkie kryteria
  - rzeczywiste problemy są bardzo często wielokryterialne.
- **Równoważenie obciążenia jest także problemem optymalizacji wielokryterialnej**
  - przy założeniach, podanych na początku prezentacji.

Pareto-optymalność

- **Pareto dominacja (dla minim.): rozwiąz. *s* dominuje nad *s′* wtw. gdy:**
  - *s* jest nie większe niż *s′* dla żadnego kryterium
  - istnieje co najmniej jedno kryterium, dla którego *s* jest mniejsze niż *s′*
- **Zbiór Pareto – zbiór (być może nieskończony) rozwiązań niezdominowanych**
  - wszystkie rozwiązania Pareto-optymalne są „równej" jakości (chyba, że wprowadzimy dodatkowe kryteria)
- **Algorytm posługujący się Pareto-optymalnością znajduje jako wynik zbiór rozwiązań**
  - z zbioru Pareto możemy wybrać jedno rozw. finalne.

Kryteria optymalizacji

- **Kryterium 1 – niezrównoważenie**

  $$\Phi_1(S) = \Sigma n \in [0,N-1] \; |NWP(S,n)/Ip(n) - WP|$$

  - średnia wartość odchylenia obciążenia procesorów zadaniami z S (znormalizowana do [0..1])

- **Kryterium 2 – komunikacja**

  $$\Phi_2(S) = \Sigma \, comExt(S) \; / \; \Sigma \, com(S)$$

  - procent komunikacji wykonywanych “zewnętrznie” w stosunku do całości komunikacji

- **Kryterium 3 – migracje**

  $$\Phi_3(S) = |\{t \in T: migrated(t)\}| \; / \; |T|$$

  - liczba migracji zadań/całkowita liczba zadań

Funkcje lokalne dla kryteriów

- **Kryterium 1 – niezrównoważenie**
  - funkcja lokalna
    $\phi_1(t) = \gamma * load(\mu_t) + (1 - \gamma) * (1 - ldev(t))$
    - $load(\mu_t)$ – miara przekroczenie śr. obciąż. przez węzeł μ_t, $ldev(t)$ – odchyl. wagi t od średniej.
- **Kryterium 2 – komunikacja**
  - funkcja lokalna
    $\phi_2(t) = 1 - attr(t)$
    - $attr(t)$ – ilość komunikacji między t a innymi zadaniami na tym samym węźle (znormalizowana w zakr. [0..1])
- **Kryterium 3 – komunikacja**
  - $\phi_3(t) = migrated(t)$

Alternatywne funkcje dla kryt. 3

- **Kryterium 3 – migracje**
  - funkcja globalna

    $\Phi_3(S) = migrFaultTotal(S) * migration\_num(S)$

    $migrFaultTotal(S) = \Sigma_{t \in T}\, migr\_qual(t)$
  - funkcja *migr_qual(t)* jest **miarą jakości migracji**
    - pokazuje jak zmienia się łączne, średnie zrównoważenie obciążenia procesorów z powodu migracji zadania *t*
- **Funkcja lokalna:**

  $\Phi_3(t) = migr\_qual(t)$

„Gradientowe” funkcje dla kryt. 1 i 2

- **Kryterium 1 – obciążenie**
  - funkcja globalna
    $\Phi_1(S) = (totalimpr(S) - 1) / 2$
    $totalimpr(S) = \Sigma_{n \in N}\, impr(n)$
  - $impr(n) = |NWP(S,n)/Ip(n) - WP| - |NWP(S^*,n)/Ip(n) - WP|$
  - funkcja $totalimpr(S) \in [-1,1]$ jest **miarą zmiany zrównoważenia**
    - pokazuje jak zmienia się łączne, średnie zrównoważenie obciążenia procesorów z powodu migracji zadań w stosunku do położenia inicjalnego
- **Kryterium 2 – komunikacja**
  $\Phi_2(S) = (extcomimpr(S) - 1) / 2$

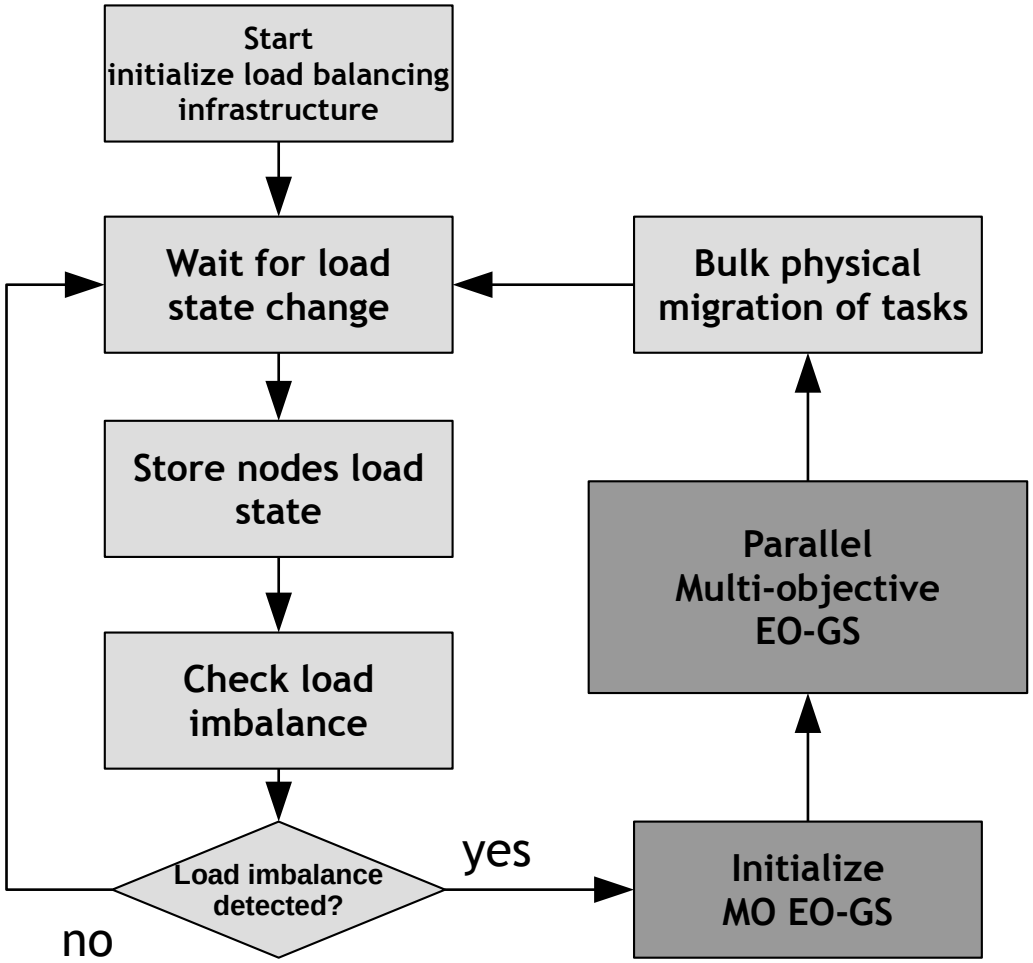
Pseudokod EO wielokryterialnego

initialize configuration S at will
$S_{\text{best}} \leftarrow S$
$D_{\text{S}} \leftarrow \emptyset$ {the set of non-dominated solutions (Pareto-front)}
while total number of iterations $\mathcal{N}_{\text{iter}}$ not reached **do**
  $C \leftarrow$ set of criteria for evaluation in the current iteration
  for all $c \in C$ **do**
    evaluate $\phi_{i,c}$ for each variable s_i of the current solution S
    rank the variables s_i based on their local fitness $\phi_{i,c}$
    choose the rank k according to $k^{-\tau}$ so that the variable s_j with $j = \pi(k)$ is selected
    evaluate ω_s for each neighbour $S_v \in \mathit{Neigh}(S, s_j)$, generated by s_j change of the current solution S
    rank neighbours $S_v \in \mathit{Neigh}(S, s_j)$ based on the target function ω_s
    choose $S' \in \mathit{Neigh}(S, s_j)$ according to the exponential distribution
    accept $S \leftarrow S'$ unconditionally
  end for
  if S is non-dominated **then**
    include S in D_{S}
  end if
end while
select S_{best} from D_{S} using $\Phi(S)$
return S_{best} and $\Phi(S_{\text{best}})$

Warianty EO wielokryterialnego

- **Kolejność optymalizacji kryteriów**
  - w trakcie działania EO, przełączanie między kryteriami następuje pseudolosowo, co ustaloną poprzez parametr liczbę iteracji
  - liczba optymalizowanych kryteriów w iteracji:
    - (1) **"jedno kryterium w zbiorze C"**
    - (2) **"wiele kryteriów w zbiorze C"**
- **Wybór rozwiązania finalnego – wg. odległości do punktu idealnego**
  - dwie miary odległości od pkt. o współrzędnych 0,0,0:
    - odległość geometryczna (euklidesowa)
    - odległość "Manhattan" (taxicab)

Schemat równoważenia MO 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.

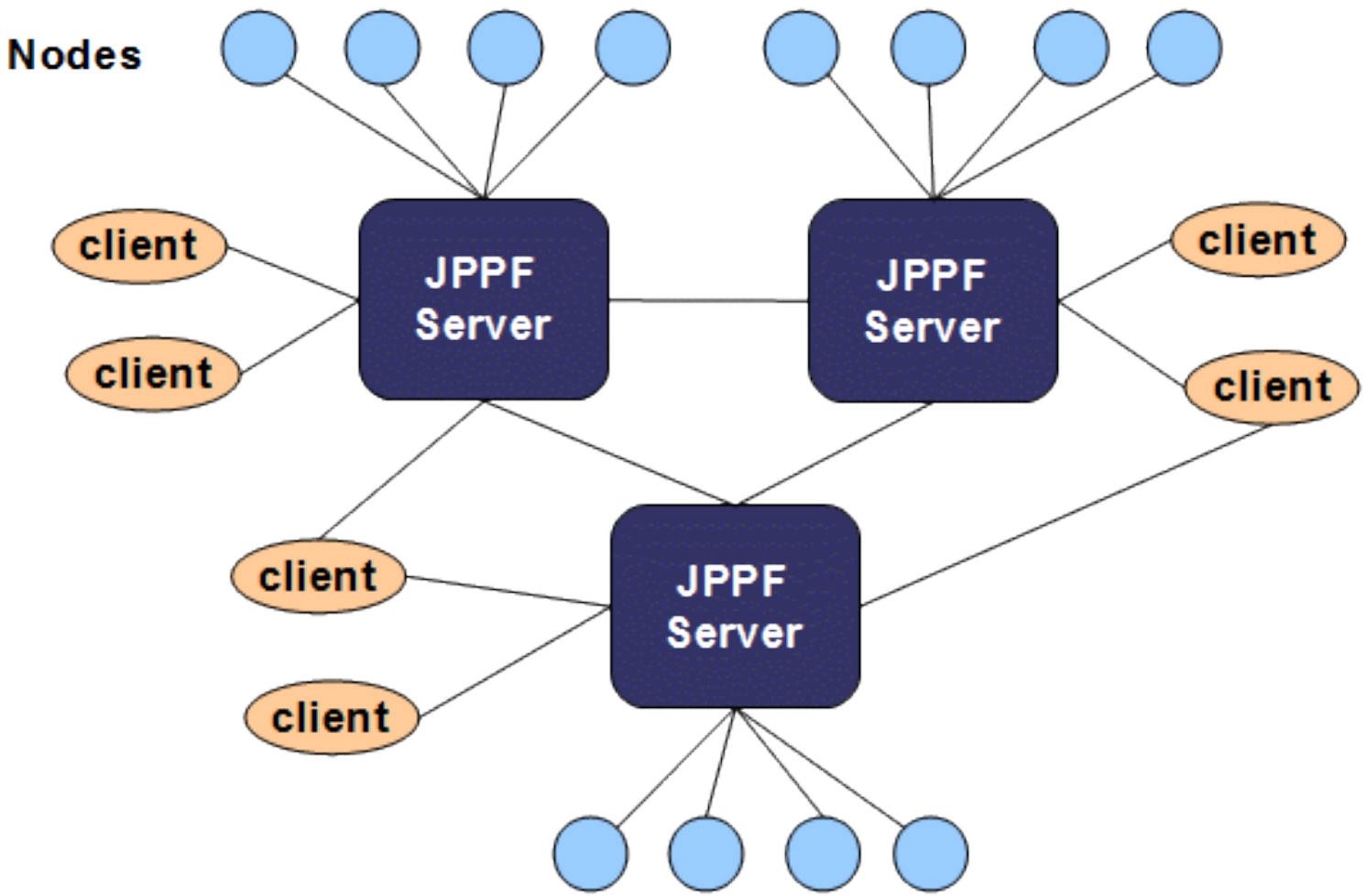
Wprowadzenie do JPPF

- **JPPF – środowisko programowe (middleware) umożliwiające uruchamianie aplikacji wymagających dużej mocy obliczeniowej na dowolnej liczbie komputerów.**
- **Realizacja**
  - podział aplikacji na mniejsze części, które mogą być wykonywane jednocześnie na różnych maszynach,
  - uruchomienie aplikacji w Gridzie JPPF.
- **Wymagania**
  - Java, Java EE, .NET, Android – Linux, Windows, MacOS X.
- ***Open source* – aktywnie rozwijany od 10 lat**
  - umożliwia uruchamianie dowolnych aplikacji (nie tylko Java),
  - modularna struktura.

Cechy JPPF

- **Dynamiczna i elastyczna architektura połączeń**
  - może zmieniać się podczas wykonywania zadań
- **Automatyczne ładowanie klas Javy (rozproszony *class loader*)**
- **Odporność na błędy, bezpieczeństwo**
  - Detekcja błędów, samo-naprawa
  - SSL/TLS encryption and authentication, data tunneling with transformation
- **Definiowanie SLA** (cechy behawioralne: filtrowanie i liczba węzłów, priorytety, zależn. czasowe, metadane definiowane przez użytkownika)
- **Wbudowane narzędzia do administracji i monitoringu**
- ***Load balancing* (API, wbudowane algorytmy oraz definiowane)**
  - **równoważenie obciążenia** – podział zleceń na podzbiory zadań.

Architektura JPPF



Równoważenie obciążenia

- **Równoważenie obciążenia** jest wykonywane zarówno na klientach JPPF, jak i na serwerach.
- JPPF używa pojęcia **kanału** jako jednostki którą równoważymy
  - dla serwera kanałem jest **połączenie** z węzłem lub z innym serwerem
  - dla klienta kanał wykonawczy jest albo **połączeniem** z serwerem lub **lokalnym wykonawcą** (przy czym klient może mieć wiele połączeń z serwerem)
  - zarówno klienci, jak i serwery mogą obsługiwać wiele zleceń jednocześnie.
- **Kategorie algorytmów**
  - **statyczny a adaptacyjny:** statyczny algorytm zawsze zwraca taki sam rozmiar pakietu dla (pary węzeł, zlecenie)
  - **lokalny a globalny:** lokalny algorytm oblicza tylko rozmiar pakietu dla pojedynczego kanału, podczas gdy globalny algorytm wylicza rozmiar pakietu na wszystkie dostępne kanały.

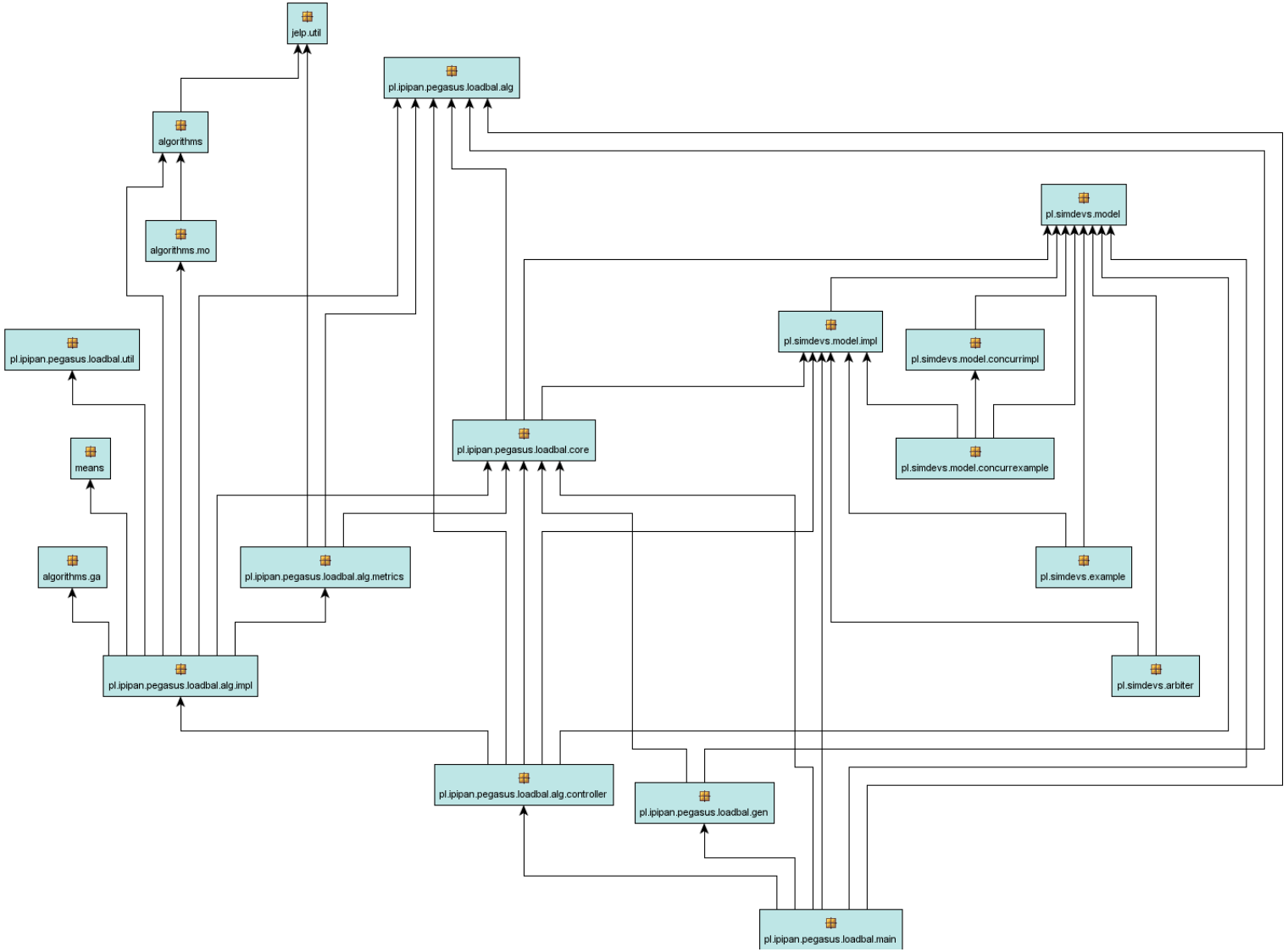
Równoważenie obciążenia – algorytmy

- **"manual"**
  - stały rozmiar paczki – Round Robin
- **"nodethreads"**
  - bundle_size = multiplicator * processing_threads
- **"autotuned"**
- **"proportional"**
- **"rl2"**

Badania eksperymentalne

- **Zastosowano 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.**

Implementacja środowiska badawczego



Implementacja środowiska badawczego cd.

AVERAGING|AveragingLoadBalance

EXTREMAL4|ExtremalLoadBalance|GUIDED_SEARCH_4|EUCLIDEAN|-|-

GENETIC|GeneticLoadBalance2

PARALLEL1|ExtremalLoadBalancePar

MO4|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U,GLOBAL_A,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

SO1|ExtremalLoadBalance|FULLY_RANDOM|EUCLIDEAN|-|-|S

SO2|ExtremalLoadBalance|GUIDED_SEARCH_2|EUCLIDEAN|-|-|S

MO42|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U,GLOBAL_A,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M_2

MO43|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U,GLOBAL_A,GLOBAL_M_2|LOCAL_U,LOCAL_A,LOCAL_M_2

MO44|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U,GLOBAL_A,GLOBAL_M_4|LOCAL_U,LOCAL_A,LOCAL_M

MO44A|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|MANHATTAN|GLOBAL_U,GLOBAL_A,GLOBAL_M_4|LOCAL_U,LOCAL_A,LOCAL_M

MO45|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U_2,GLOBAL_A,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

MO45A|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|MANHATTAN|GLOBAL_U_2,GLOBAL_A,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

EXTREMAL4G|ExtremalLoadBalance|GUIDED_SEARCH_4|EUCLIDEAN|-|-|G

MO4A|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|MANHATTAN|GLOBAL_U,GLOBAL_A,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

MO46|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U,GLOBAL_A,GLOBAL_M,GLOBAL_U_2|LOCAL_U,LOCAL_A,LOCAL_M

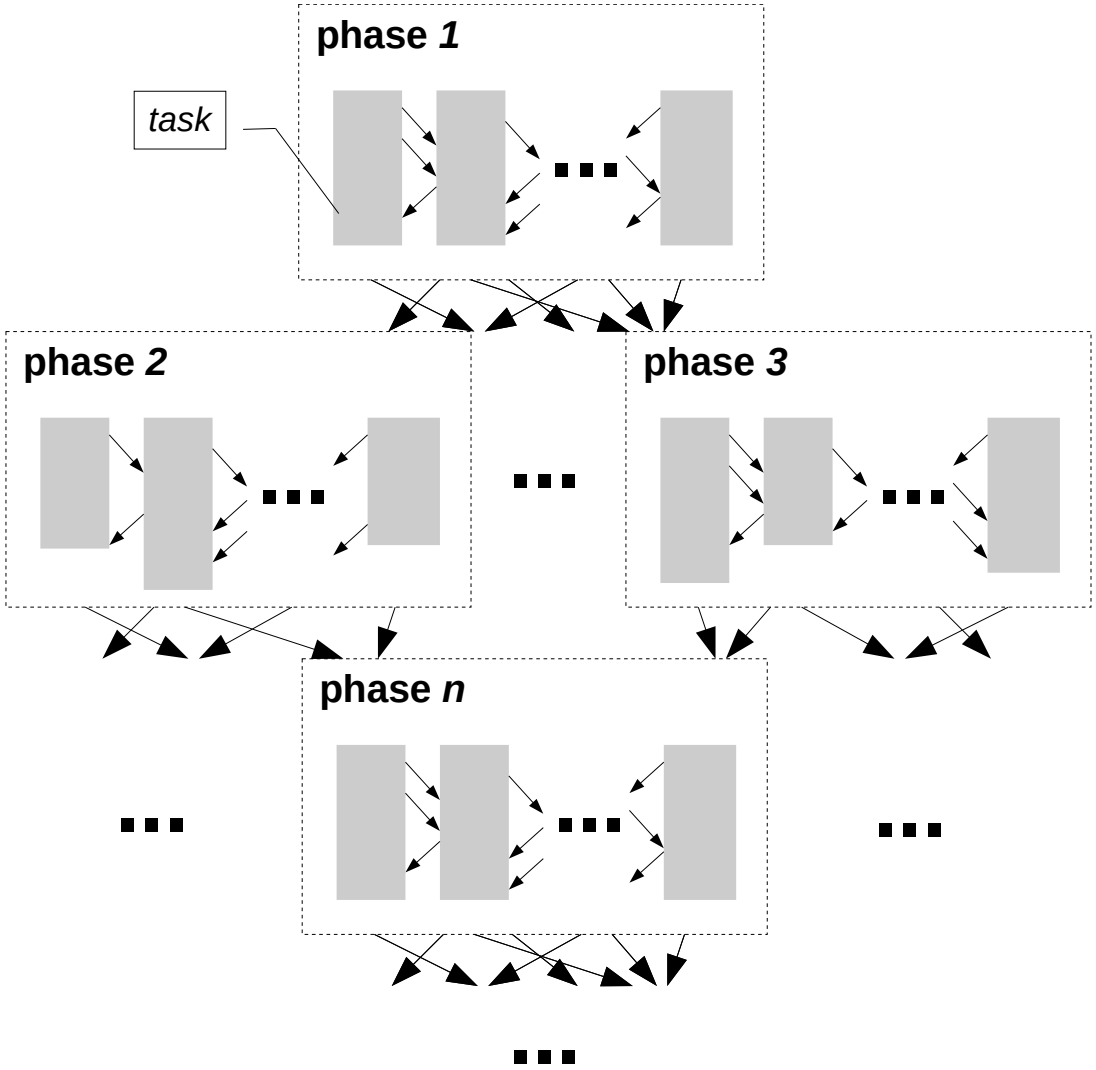
MO46A|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|MANHATTAN|GLOBAL_U,GLOBAL_A,GLOBAL_M,GLOBAL_U_2|LOCAL_U,LOCAL_A,LOCAL_M

MO47|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|EUCLIDEAN|GLOBAL_U_2,GLOBAL_A_2,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

MO47A|ExtremalLoadBalanceMo|GUIDED_SEARCH_4|MANHATTAN|GLOBAL_U_2,GLOBAL_A_2,GLOBAL_M|LOCAL_U,LOCAL_A,LOCAL_M

PA|ExtremalParallelRunner

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].
- **Rozmiar systemu wykonawczego:**
  - 2-32 /małe aplikacje/, 32-128 /duże aplikacje/.
- **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 $\lambda = 0.5$
  - U5-U9 – zestawy wartości β, γ, Δ_1, Δ_2
  - waga migracji: M0 (0%), M1 (20%), M2 (40%)
- **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 $N_{iter} = 500$.**

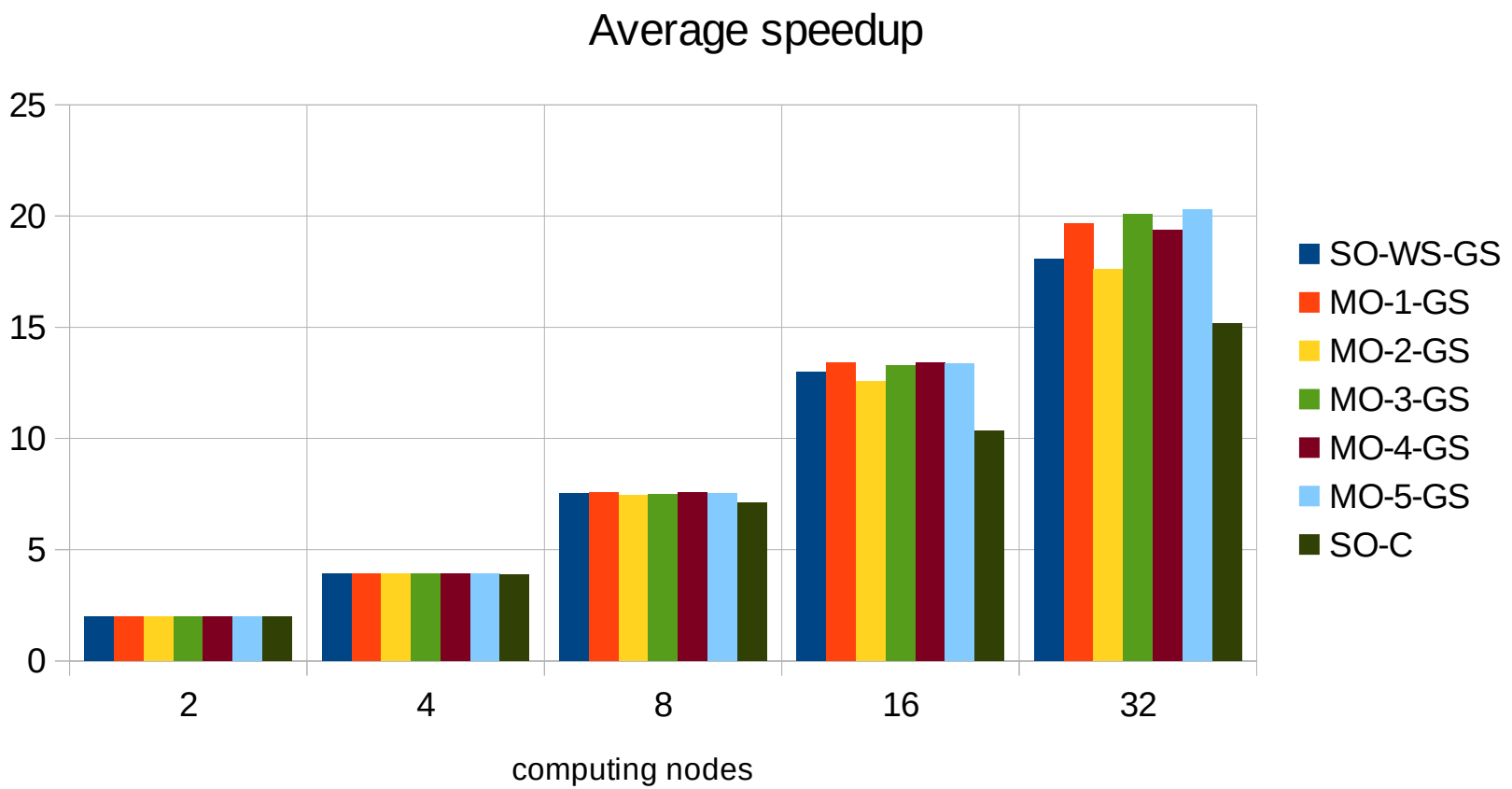
Badania eksperymentalne cd.

- **Co badaliśmy (wcześniej):**
  - EO, EO-GS /EO4/
  - PEO-A, PEO-GS-A, PEO-GS-B
  - PEO-GS-C, -D, -E
  - EO4G
  - MO1, MO1S, MO2, MO2A, MO2AS, MO2S
  - SO1, SO2
- **Badania porównawcze:**
  - GA, DT

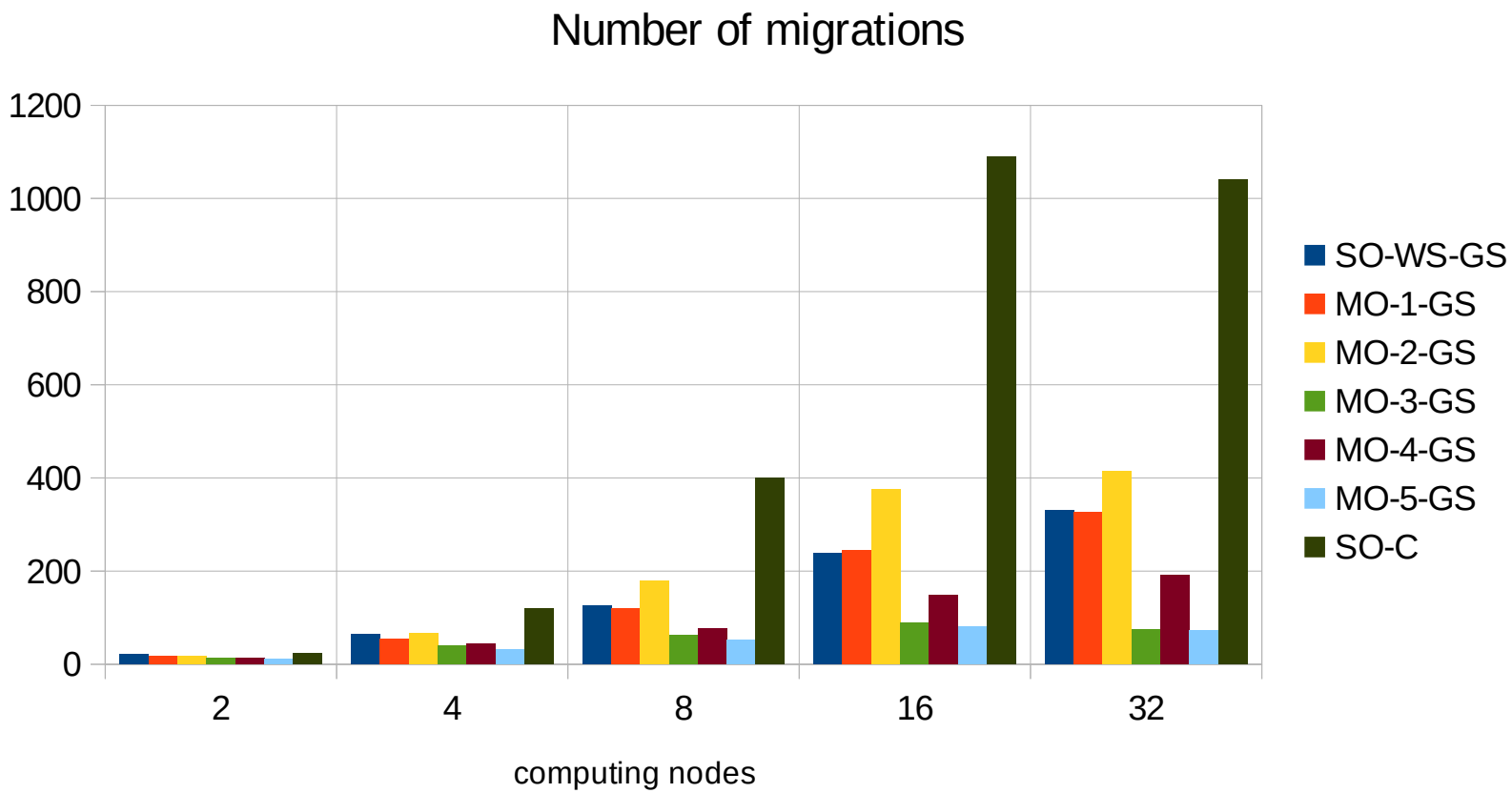
Badania eksperymentalne – wyniki prezentowane

- **Algorytmy wielokryterialne:**
  - MO-1-GS (kryteria „klasyczne”),
  - MO-2-GS (alternatywna formuła dla kryt. 3)
  - MO-3-GS (kryteria 1 “gradientowej”)
  - MO-4-GS (kryt. 1 w formule “gradientowej” oraz czwarte alt. kryt.)
  - MO-5-GS (kryteria 1 i 2 w formule “gradientowej”)
- **Algorytmy referencyjne:**
  - EO-GS /SO-WS-GS/
  - SO1 /SO-C/

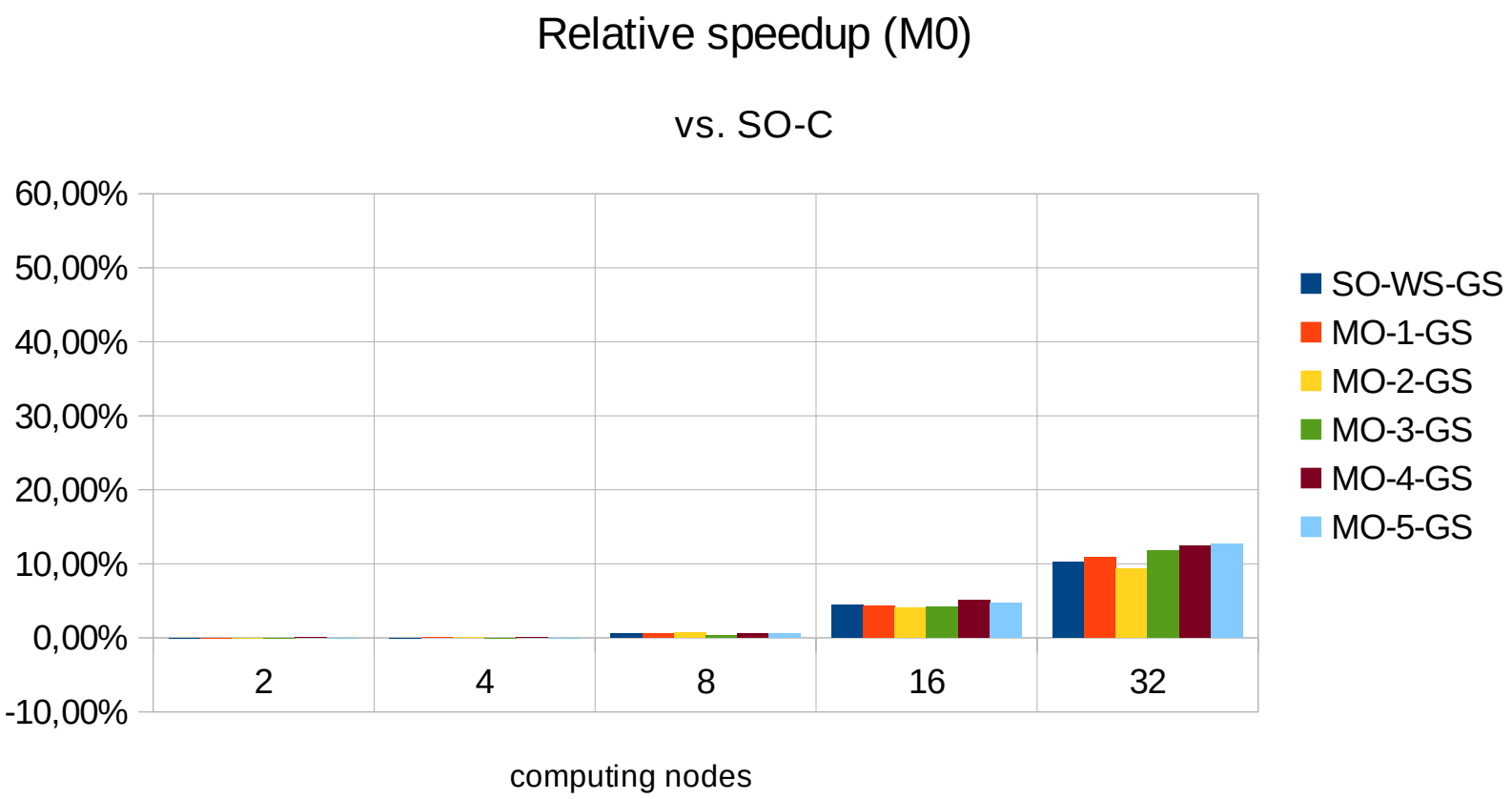
Przyśpieszenie



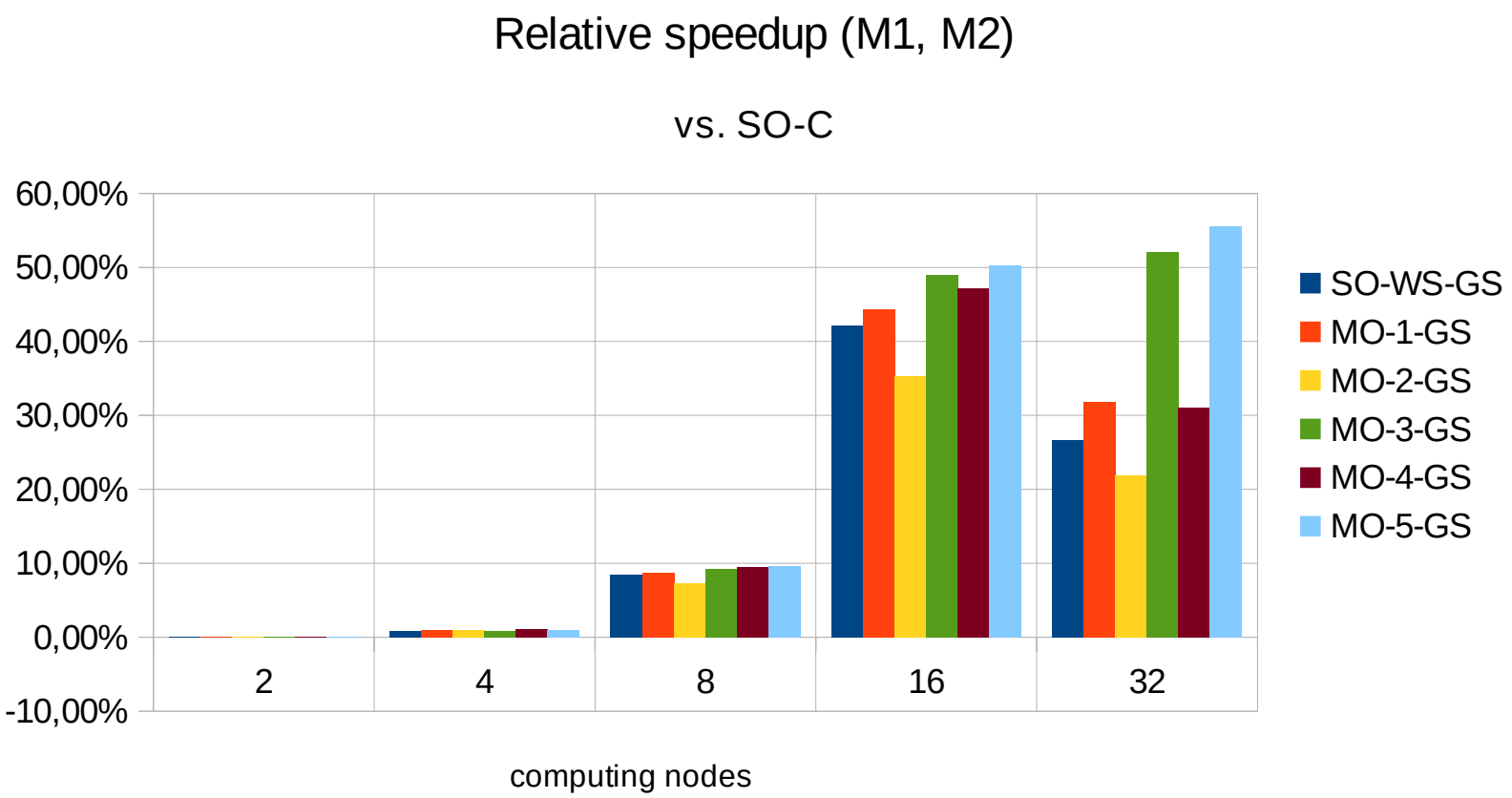
Migracje


Number of migrations
1200
1000
800
600
400
200
0
2
4
8
16
32
computing nodes
SO-WS-GS
MO-1-GS
MO-2-GS
MO-3-GS
MO-4-GS
MO-5-GS
SO-C

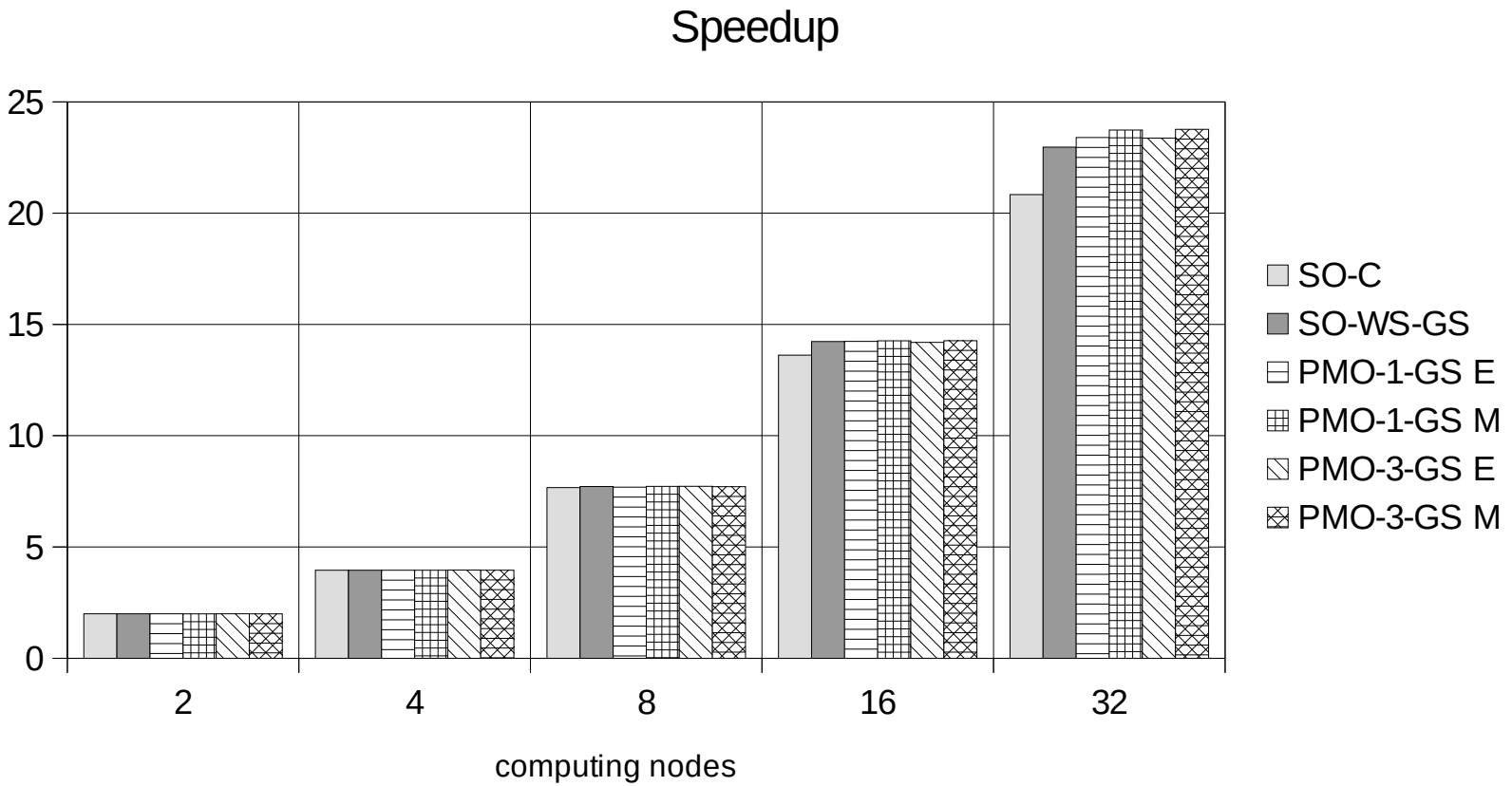
Poprawa przyśpieszenie (średnia)



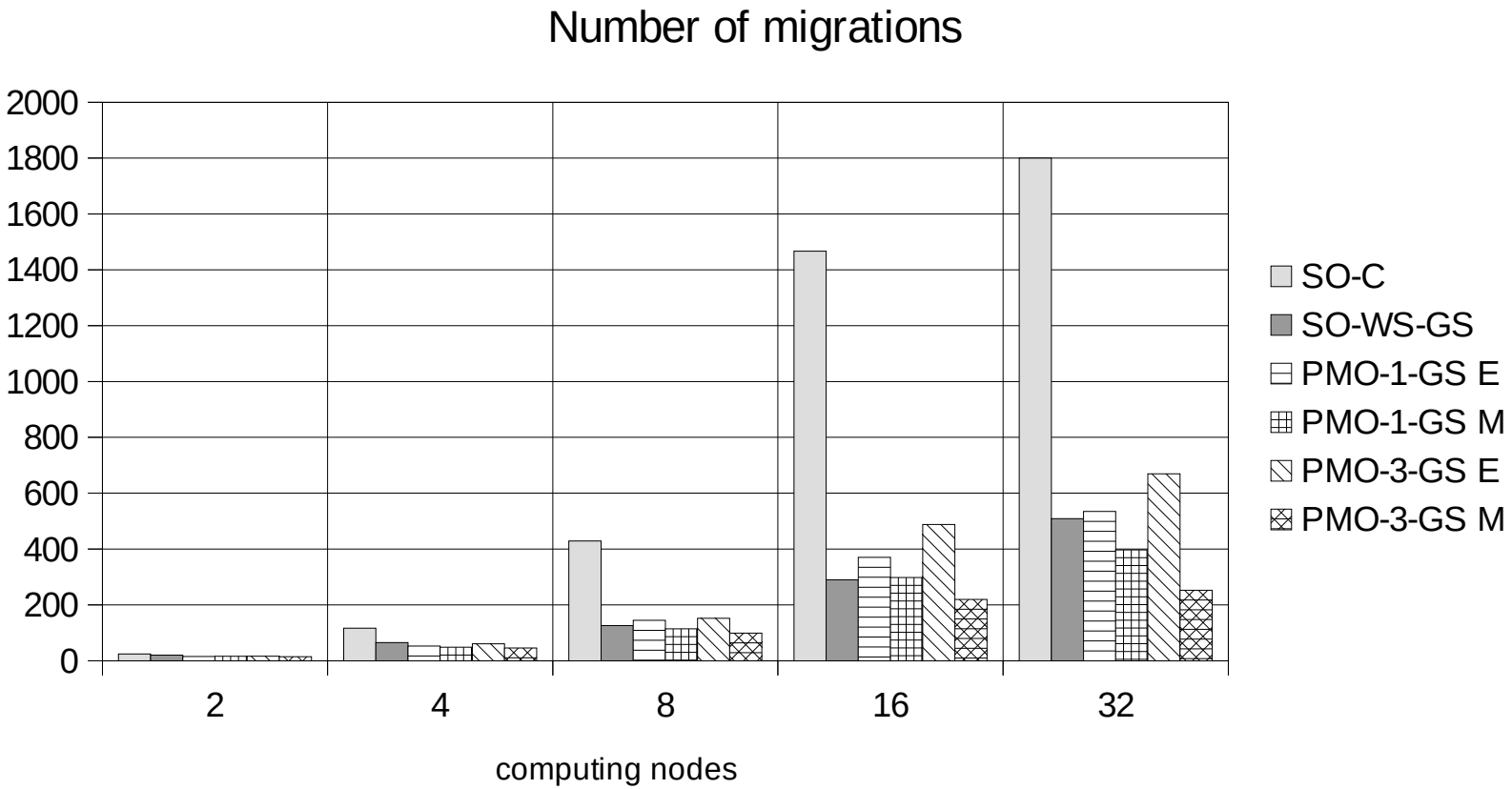
Poprawa przyśpieszenie (średnia)



Przyśpieszenie dla różnych miar odległości

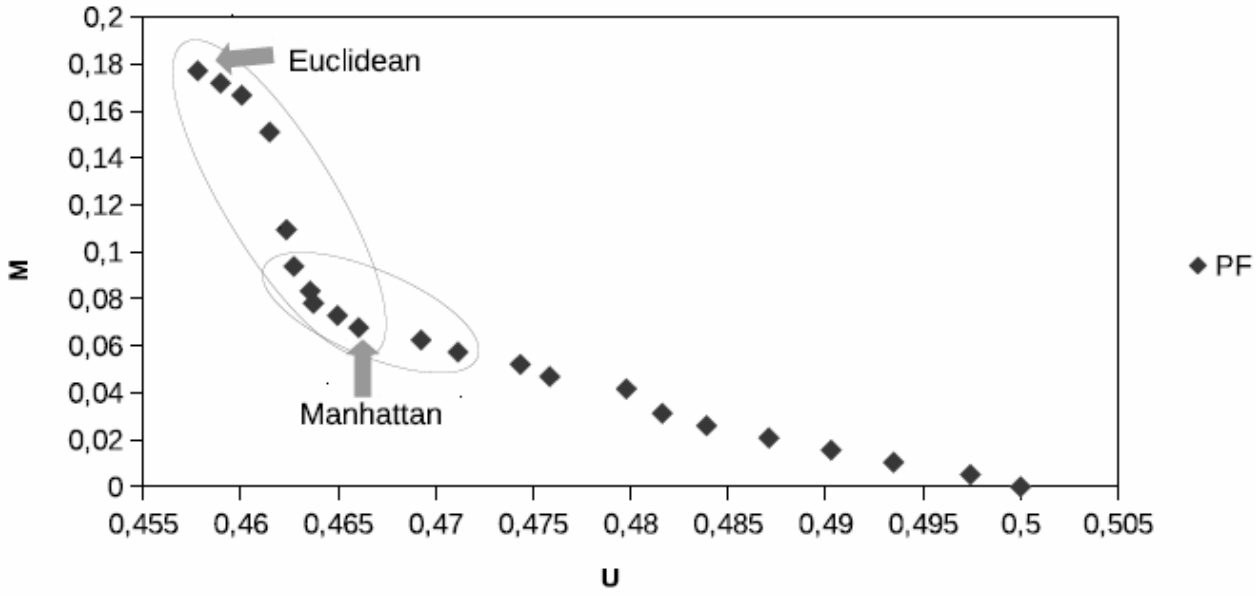


Migracje dla różnych miar odległości

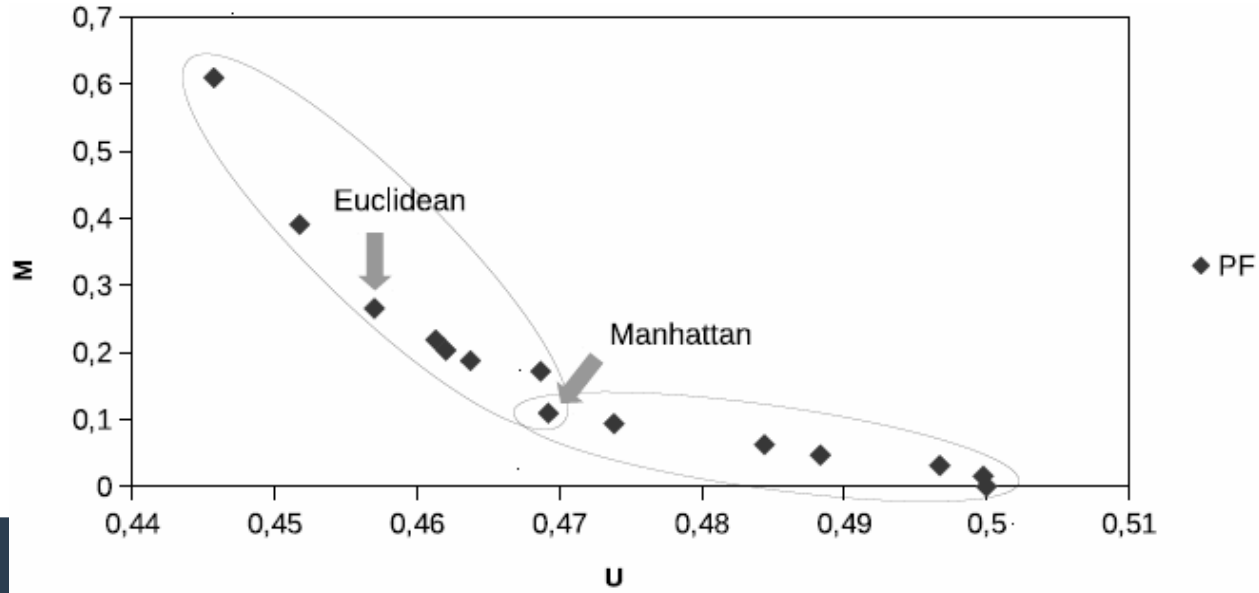


Pareto front dla różnych miar odległości

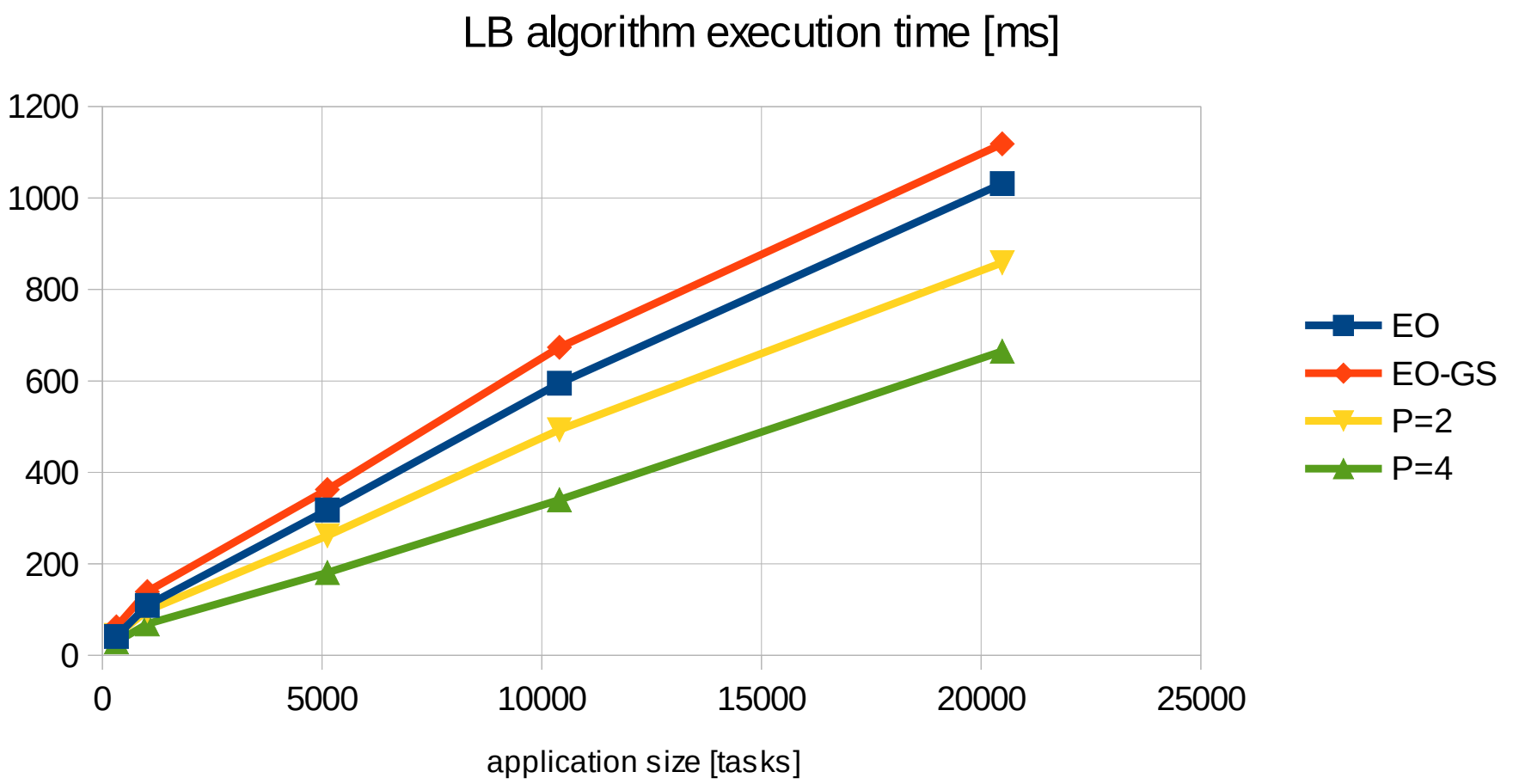
- PMO-3-GS, P9F, 32 węzły



- PMO-3-GS, P9A, 32 węzły



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!