Wacław KUŚ

Katedra Wytrzymałości Materiałowych i Metod Komputerowych Mechaniki
Wydział Mechaniczny Technologiczny, Politechnika Śląska, Gliwice

GRIDY OBLICZENIOWE W ZAGADNIENIACH OPTYMALIZACJI I IDENTYFIKACJI UKŁADÓW MECHANICZNYCH

Streszczenie. W artykule przedstawiono metodę optymalizacji i identyfikacji układów mechanicznych z wykorzystaniem algorytmów ewolucyjnych i gridów obliczeniowych. Przedstawiono przykłady numeryczne uzyskane z wykorzystaniem środowisk Condor oraz UNICORE.

COMPUTATIONAL GRIDS IN OPTIMIZATION AND IDENTIFICATION PROBLEMS OF MECHANICAL STRUCTURES

Summary. The optimization and identification of mechanical structures based on evolutionary algorithms and computational grids are presented in the paper. The numerical examples obtained using Condor and UNICORE environments are shown.

1. Wstęp

Zagadnienia optymalizacji konstrukcji należą do zadań trudnych i czasochłonnym obliczeniowo. Wynika to zarówno z potrzeby zagwarantowania optimum globalnego rozwiązywanego zadania, jak i konieczności wielokrotnego rozwiązywania złożonych fizycznie (sprężystość, plastyczność,) i geometrycznie zadań bezpośrednich, w postaci zagadnień brzegowych lub brzegowo-początkowych fizyki matematycznej, służących do oceny funkcji celu. Przeprowadzenie optymalizacji niejednokrotnie wymaga dostępu do wysoko wydajnych superkomputerów lub klastrów obliczeniowych. Optymalizację przeprowadza się zazwyczaj stosując metody klasyczne (najczęściej lokalne), oparte na znajomości gradientu funkcji celu, lub metody ewolucyjne (globalne) [4,63,50,69]. Z prac badawczych prowadzonych w Katedrze Wytrzymałości Materiałów i Metod Komputerowych Mechaniki [10,11,12,13,14,16,21,22,42,60,61,71] wynika, że zastosowanie metod klasycznych, ewolucyjnych, jak również połączenie obu tych metod (metody hybrydowe) [12,13,15,19,66] pozwala na skuteczną optymalizację konstrukcji. Opracowano również oprogramowanie pozwalające na sprawdzenie skuteczności równoległych i rozproszonych algorytmów

ewolucyjnych. Na podstawie wyników badań [20,24,25,26,27,28,29,30,31,32,33,34,35,38,59] można wnioskować, że zastosowanie tych algorytmów pozwala na znaczne skrócenie czasu optymalizacji lub też zwiększenie poziomu komplikacji zadań przy nie zmienionym czasie optymalizacji.

W artykule przedstawiono przykłady wykorzystania środowisk gridowych w optymalizacji i identyfikacji układów mechanicznych. Zastosowano rozproszony algorytm ewolucyjny oraz systemy Condor [40] i UNICORE [74]. Obliczenia przeprowadzone zostały z wykorzystaniem sprzętu komputerowego Katedry Wytrzymałości Materiałów i Metod Komputerowych Mechaniki. Określenie wartości funkcji celu dokonywane było z użyciem metody elementów skończonych (MES) [56].

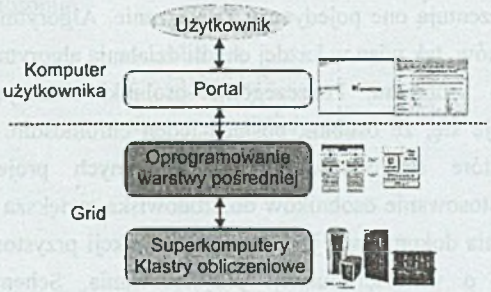
2. Gridy obliczeniowe

Grid obliczeniowy [47,46,54,57] jest grupą zasobów komputerowych (superkomputerów, klastrów obliczeniowych, pamięci masowych, oprogramowania) połączonych infrastrukturą sieciową. Najważniejsze cechy gridów to: udostępnienie dużej mocy obliczeniowej poprzez efektywne wykorzystanie rozproszonych zasobów, zapewnienie bezpieczeństwa zasobów, jednokrotne logowanie w celu uzyskania dostępu do zasobów. Prace nad rozwojem gridów prowadzone są w ramach wielu projektów europejskich [6,41,43,44,53], amerykańskich [49,72] oraz azjatyckich [64,65]. Również polskie jednostki naukowe biorą udział w wielu projektach europejskich [2,5,6,7,41,43,44,51,52] oraz krajowych [39,67,70,75].

W systemach gridowych można wyróżnić kilka głównych składników (rys. 1). Portal jest często jedynym składnikiem systemu gridowego, z którym ma do czynienia użytkownik w trakcie rozwiązywania zadań. Portal jest implementowany zazwyczaj jako zaawansowana strona internetowa lub prosty w obsłudze program [3,58]. Portale umożliwiają użytkownikom, nie znającym obsługi wielu różnych systemów superkomputerów oraz klastrów obliczeniowych, na uruchamianie skomplikowanych zadań. Portale tworzone są w taki sposób, aby uruchomienie zadania korzystającego z różnych zasobów komputerowych przeprowadzane było w jak najprostszy sposób. Użytkownik końcowy oprogramowania nie jest zmuszany do poznania systemów operacyjnych, systemów kolejkowania oraz szczegółów obsługi superkomputerów oraz klastrów obliczeniowych, co pozwala na skupienie się na poprawnej budowie rozwiązywanego zadania oraz analizie wyników. Oprogramowanie warstwy pośredniej (*middleware*) zapewnia mechanizmy weryfikacji użytkowników, przydzielanie zasobów komputerowych oraz komunikację z docelowymi superkomputerami oraz klastrami obliczeniowymi. Oprogramowanie to tworzone jest tak, aby maksymalnie wykorzystać dostępne cechy systemów operacyjnych oraz najpopularniejsze programy kolejkowania zadań na docelowych komputerach.

Ostatnią częścią składową systemu gridowego są superkomputery oraz klastry obliczeniowe. Ze względu na częste wykorzystywanie wielu superkomputerów lub klastrów

obliczeniowych w ramach jednego zadania ważne są wysoko wydajne połączenia siecią komputerową maszyn [55].



Rys. 1. Grid obliczeniowy
Fig. 1. Computational grid

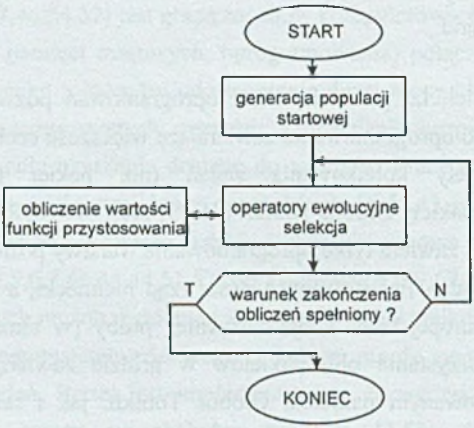
W ciągu ostatnich lat powstało wiele oprogramowań pozwalających tworzyć gridy. Początkowo tworzono oprogramowanie zawierające większość cech obliczeń gridowych jako zaawansowane pakiety kolejkowania zadań (np. pakiet Condor [40]). Obecnie najpopularniejsze są pakiety Globus Toolkit [45,49] oraz UNICORE [3,68,73,74]. Pierwszy z nich powstał w USA i zawiera tylko oprogramowanie warstwy pośredniej. System UNICORE powstał w ramach badań finansowanych przez rząd niemiecki, a obecnie rozwijany jest w ramach projektów europejskich. Podjęto również próby (w ramach projektu GRIP [53]) umożliwiające wykorzystanie obu pakietów w gridzie zawierającym zarówno serwery pracujące z zainstalowanym pakietem Globus Toolkit, jak i serwery zawierające pakiet UNICORE. Również pakiet Condor (Condor-G) rozszerzono o możliwość współpracy z gridami opartymi na pakiecie Globus Toolkit. W celu standaryzacji rozwiązań gridowych powołano Global Grid Forum [48].

Równolegle do rozwoju pakietów gridowych przeprowadzane są badania nad zastosowaniem gridów w wielu dziedzinach nauki. Pierwsze obszary, w których stosuje się systemy gridowe, to: modelowanie zjawisk meteorologicznych, obliczenia chemiczne oraz biologiczne [6,62].

Same gridy nie pozwalają na przeprowadzenie optymalizacji konstrukcji. Konieczne jest uzupełnienie ich odpowiednimi portalami oraz oprogramowaniem superkomputerów oraz klastrów obliczeniowych. W ramach projektu Eurogrid [44] opracowywane są koncepcje pozwalające na rozwiązywanie zadań analizy z wykorzystaniem oprogramowania metody elementów skończonych dla ciał stałych (Nastran) i mechaniki płynów (Fluent). Parametryczną optymalizację można przeprowadzić używając systemu Nimrod-G[64].

3. Rozproszony algorytm ewolucyjny

Algorytmy ewolucyjne [4] są algorytmami przeszukującymi przestrzeń rozwiązań, bazującymi na analogii do biologicznej ewolucji gatunków. Podobnie jak w biologii mówimy o osobnikach, reprezentują one pojedyncze rozwiązanie. Algorytmy ewolucyjne operują na populacjach osobników, tak więc w każdej chwili działania algorytmu mamy do czynienia ze zbiorem rozwiązań problemu. Poszczególne osobniki składają się z chromosomów. Zazwyczaj przyjmuje się, że osobnik posiada jeden chromosom. W skład chromosomów wchodzą geny, które są odpowiednikami zmiennych projektowych w zadaniach optymalizacji. Przystosowanie osobników do środowiska zwiększa szansę jego przetrwania. Ocena przystosowania dokonywana jest z użyciem funkcji przystosowania. Wszystkie geny osobnika decydują o wartości funkcji przystosowania. Schemat działania algorytmu ewolucyjnego przedstawiony jest na rys. 2.



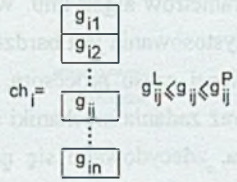
Rys. 2. Schemat działania algorytmu ewolucyjnego
Fig. 2. The evolutionary algorithm flowchart

W pierwszym kroku tworzona jest populacja startowa osobników. Tworzy się ją zazwyczaj losując wartości genów poszczególnych osobników. Kolejny etap polega na określeniu wartości funkcji przystosowania osobników. Następnie operatory ewolucyjne zmieniają geny osobników populacji rodziców, dokonywana jest selekcja osobników, które znajdą się w populacji potomnej. Populacja potomna staje się populacją rodziców i algorytm wykonywany jest iteracyjnie aż do spełnienia warunku zakończenia obliczeń. Warunek zakończenia obliczeń jest najczęściej formułowany jako maksymalna liczba iteracji.

W algorytmach ewolucyjnych stosuje się reprezentację zmiennoprzecinkową, tzn. geny zawarte w chromosomach zawierają liczby zmiennoprzecinkowe. Zazwyczaj stosuje się ograniczenia na zmienność wartości genów.

Osobnik jednochromosomowy (tzn. chromosom) ch_i, $i=1,2,..,N$, gdzie N jest rozmiarem populacji, można przedstawić za pomocą macierzy kolumnowej lub wierszowej, której elementy reprezentowane są przez geny g_{ij}, $j=1,2,...,n$,

n- liczba genów w chromosomie.



Rys. 3. Geny chromosomu

Fig. 3. The genes of the chromosome

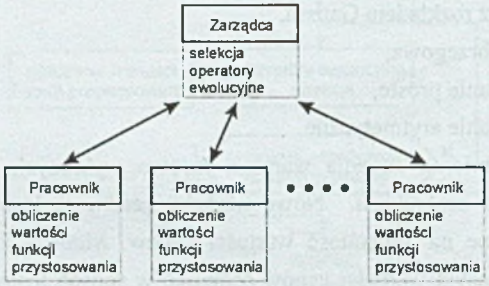
Operatory ewolucyjne zmieniają wartości genów podobnie jak biologiczne mechanizmy mutacji i krzyżowania. W literaturze opisanych jest wiele operatorów ewolucyjnych. Podstawowe z nich to:

- mutacja równomierna,
- mutacja z rozkładem Gaussa,
- mutacja brzegowa,
- krzyżowanie proste,
- krzyżowanie arytmetyczne.

Mutacja równomierna zmienia wartości losowo wybranych genów w przypadkowo wyselekcjonowanych osobnikach. Nowe wartości genów są losowane, tak aby spełniały ograniczenia nałożone na zmienność wartości genów. Mutacja z rozkładem Gaussa jest operatorem zmieniającym wartości genów osobnika w sposób losowy, podobnie do mutacji równomiernej. Nowe wartości genów powstają z wykorzystaniem liczb losowych otrzymanych z rozkładu Gaussa. Operator ten przeszukuje otoczenie osobnika. Mutacja brzegowa działa w sposób podobny do mutacji równomiernej, z tym że nowe wartości, które otrzymują geny, są równe lewemu lub prawemu ograniczeniu na zmienność genu. Krzyżowanie proste jest operatorem tworzącym osobnika potomnego na podstawie dwóch osobników rodzicielskich. Losowana jest pozycja cięcia, nowy osobnik składa się z części genów pierwszego oraz części drugiego osobnika. Krzyżowanie arytmetyczne nie ma swojego odpowiednika biologicznego. Nowy osobnik powstaje podobnie jak w przypadku krzyżowania prostego na podstawie dwu osobników rodzicielskich, przy czym wartości genów osobnika określane są jako średnia wartości genów osobników rodzicielskich. Ważnym składnikiem algorytmu ewolucyjnego jest mechanizm selekcji. Prawdopodobieństwo przetrwania osobnika zależy od wartości funkcji przystosowania. Selekcja rangowa wykonywana jest w kilku etapach. W pierwszej kolejności sortujemy osobniki biorąc pod uwagę wartości funkcji przystosowania, następnie każdemu z osobników przypisujemy wartość rangi. Zależy ona od numeru osobnika oraz od funkcji rangi. Najlepsze osobniki otrzymują największą wartość

rangi, najgorsze najmniejszą. Ostatni etap polega na wylosowaniu osobników do pokolenia potomnego, przy czym prawdopodobieństwo wylosowania poszczególnych osobników jest ściśle związane z wartością rangi osobników. Zwiększenie szybkości działania algorytmu ewolucyjnego w większości przypadków związane jest ze skróceniem czasu obliczeń funkcji celu i odpowiednim doborem parametrów algorytmu. W przypadku zadań optymalizacji, dla których czas obliczeń funkcji przystosowania jest bardzo krótki, a zastosowanie operatorów i realizacja selekcji zajmują najwięcej czasu procesora, istotne jest zoptymalizowanie kodu algorytmu ewolucyjnego. Ponieważ zadania mechaniki są zadaniami, które pochłaniają dużo czasu obliczeniowego komputera, zdecydowano się na skrócenie czasu obliczeń poprzez skrócenie czasu obliczeń funkcji przystosowania oraz dobór parametrów algorytmu ewolucyjnego.

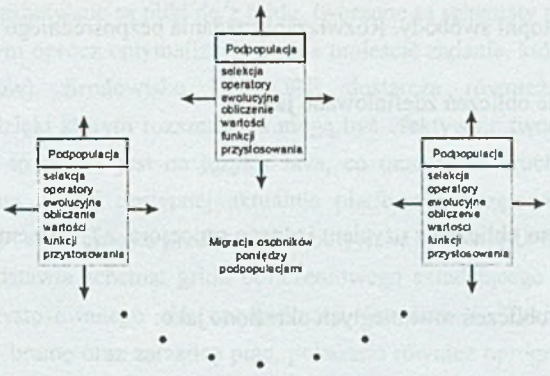
W celu zwiększenia szybkości działania algorytmu ewolucyjnego, szczególnie w przypadku stosowania komputerów wieloprocesorowych czy też clustrów obliczeniowych, stosuje się algorytmy równoległe (ang. parallel), rozproszone (ang. distributed). Równoległe algorytmy ewolucyjne mają architekturę master-worker (rys. 4). Operują one podobnie jak algorytmy sekwencyjne na pojedynczej populacji osobników.



Rys. 4. Równoległy algorytm ewolucyjny
Fig. 4. The parallel evolutionary algorithm

Zarządca dokonuje selekcji oraz wykonuje operatory genetyczne, pracownicy natomiast obliczają wartości funkcji przystosowania poszczególnych osobników przesyłanych przez zarządcę. Maksymalne przyspieszenie obliczeń (liniowe), które może zostać osiągnięte w przypadku równoległych algorytmów ewolucyjnych, jest równe n.

Rozproszone algorytmy ewolucyjne bazują na teorii algorytmów koewolucyjnych. Proces ewolucji przebiega szybciej, jeśli ewoluują odosobnione podpopulacje o małej liczbie osobników wymienianych pomiędzy sobą. W algorytmie tym populację osobników dzieli się na kilka lub kilkanaście podpopulacji. Każda z podpopulacji ewoluuje oddzielnie i tylko co pewien czas następuje faza migracji, podczas której część osobników wymieniana jest pomiędzy podpopulacjami. Schemat algorytmu rozproszonego przedstawiony jest na rys. 5.



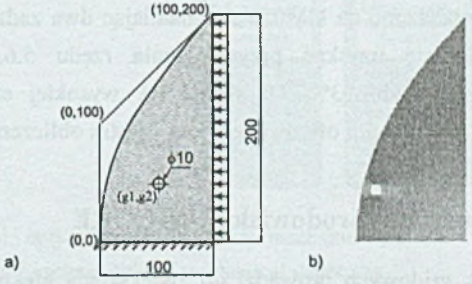
Rys. 5. Rozproszony algorytm ewolucyjny
Fig. 5. The distributed evolutionary algorithm

4. Optymalizacja z użyciem systemu Condor

Badania autorów [36], przeprowadzone z wykorzystaniem systemu Condor oraz klastra wyposażonego w 6 procesorów, wskazują na celowość dalszych badań w kierunku obliczeń gridowych. Przykładowy test dotyczy optymalizacji tarczy z otworem w kształcie koła (rys. 6). Zadanie optymalizacji polegało na takim doborze współrzędnych środka otworu, aby zminimalizować funkcjonał naprężeniowy:

$$F(x) = \int_{\Omega} \sigma_{eq} \, d\Omega , \tag{1}$$

gdzie σ_{eq} jest naprężeniem redukowanym.



Rys. 6. a) Geometria tarczy, b) najlepszy wynik – mapa naprężeń
Fig. 6. a)The plate geomery, b) the best result – stresses map

Optymalizację przeprowadzono z wykorzystaniem algorytmu koewolucyjnego (rozproszonego algorytmu ewolucyjnego). Pojedyncze zadanie rozwiązywane było z wykorzystaniem MES. Rozpatrywana tarcza były gęsto siatkowana, rozmiar zadania wynosił

około 50 tysięcy stopni swobody. Rozwiązanie zadania bezpośredniego zajmowało około 450 sekund.

Przyspieszenie obliczeń zdefiniowano jako:

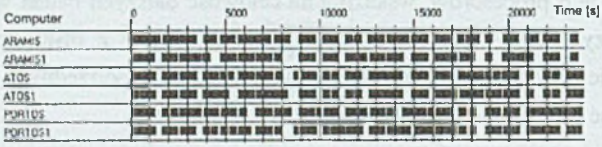
$$s = \frac{T_1}{T_n}, \tag{2}$$

gdzie T_1 jest czasem obliczeń z użyciem jednego procesora, a T_n czasem obliczeń przy użyciu n procesorów.

Efektywność obliczeń równoległych określono jako:

$$E = \frac{s}{n}, \tag{3}$$

Rozwiązano dwa zadania optymalizacji. Pierwsze zadanie rozwiązano na dedykowanym klastrze. Uzyskano przyspieszenia dla 6 procesorów rzędu 4.04, a efektywność zrównoleglenia obliczeń rzędu 67%. Na rysunku 7 przedstawiono wykorzystanie poszczególnych procesorów klastra podczas 20 pierwszych pokoleń działania algorytmu ewolucyjnego.



Rys. 7. Wykorzystanie poszczególnych procesorów w ciągu 20 pierwszych pokoleń algorytmu ewolucyjnego

Fig. 7. The use of processors in first 20 generations of evolutionary algorithm

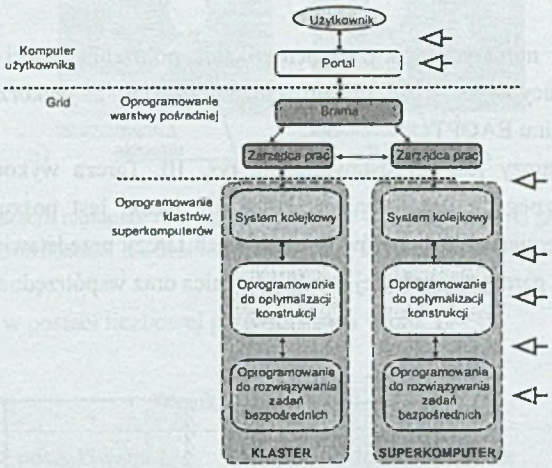
Drugi test przeprowadzono na klastrze uruchamiając dwa zadania optymalizacji w tym samym czasie. Udało się uzyskać przyspieszenia rzędu 5.6, co daje efektywność zrównoleglenia obliczeń rzędu 93%. Uzyskanie tak wysokiej efektywności świadczy o możliwości wykorzystania w pełni oferowanej mocy klastra obliczeniowego.

5. Identyfikacja z użyciem środowiska UNICORE

Użycie środowisk gridowych prowadzi do zwiększenia efektywności użycia zasobów sprzętowych i programowych. Powstanie środowisk takich jak UNICORE pozwala również znacznie uprościć proces przygotowywania zadania optymalizacji, doboru parametrów. Nawet użytkownik nie znający środowiska docelowego, na którym będzie przeprowadzana optymalizacja, może skorzystać z zasobów superkomputerowych dzięki odpowiedniemu oprogramowaniu, portalom gridowym. Środowisko UNICORE zapewnia standaryzację wymiany informacji pomiędzy zasobami gridowymi a komputerem użytkownika. Za pomocą UNICORE przeprowadzana jest autoryzacja użytkownika (z użyciem bezpiecznych

certyfikatów X.509), przesyłane są pliki do/z gridu, tworzone są schematy prac do wykonania (np. diagram, w którym oprócz optymalizacji można umieścić zadanie, którego celem będzie wizualizacja wyników). Środowisko UNICORE dostarcza również wielu narzędzi programistycznych, dzięki którym rozszerzenia mogą być efektywnie tworzone w oparciu o pluginy. Środowisko to oparte jest na języku Java, co umożliwia uruchomienie aplikacji klienta praktycznie na każdej dostępnej aktualnie platformie sprzętowej (począwszy od Windows, Linux, BSD, a kończąc na platformach opartych na systemie UNIX).

Rysunek 8 przedstawia schemat gridu obliczeniowego składającego się z klastra oraz superkomputera, przystosowanego do optymalizacji konstrukcji. Zaznaczono składniki warstwy pośredniej – bramę oraz zarządcę prac, pokazano również oprogramowanie służące do kolejkowania zadań na docelowych komputerach. Oprogramowanie do optymalizacji konstrukcji umieszczono na klastrze i superkomputerze. Podczas optymalizacji najwięcej czasu zajmuje rozwiązywanie zadań bezpośrednich, oprogramowanie używane w tym celu również znajduje się na klastrze oraz superkomputerze. Strzałki pionowe określają kierunki komunikacji pomiędzy składnikami grida. Strzałkami po prawej stronie rysunku zaznaczono obszary, które są rozwijane i adaptowane w Katedrze Wytrzymałości Materiałów i Metod Komputerowych Mechaniki.

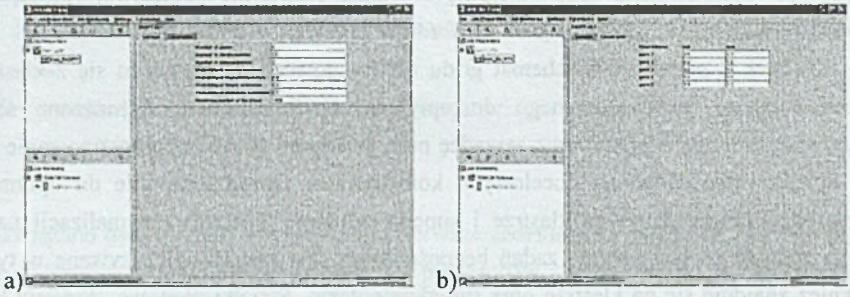


Rys. 8. Grid obliczeniowy do optymalizacji konstrukcji mechanicznych
Fig. 8. Computational grid for optimization of mechanical structures

W artykule przedstawiono plugin EAOPT do systemu UNICORE, który umożliwia wprowadzenie parametrów algorytmu ewolucyjnego, jak również wysłanie tych parametrów i przeprowadzenie optymalizacji przez zasoby gridowe. Oprócz pluginu powstał program optymalizacji ewolucyjnej dla systemu Linux. Program ten umieszczany jest na platformie gridowej.

Rysunek 9 przedstawia wygląd okna klienta UNICORE z uruchomionym pluginem EAOPT.

W przyszłości planowane jest rozszerzenie funkcjonalności plugina o wizualizację wyników oraz wizualizację przebiegu optymalizacji.

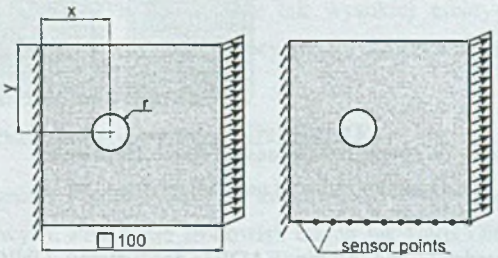


Rys.9. Ekran klienta UNICORE z uruchomionym pluginem EAOPT: a) parametry AE, b) ograniczenia na zmienne

Fig. 9. The client appliation of the UNICORE with EAOPT plugin: a) EA parametres, b) project variables constraints

Celem testu numerycznego jest identyfikacja położenia pustki w tarczy z użyciem algorytmu ewolucyjnego. Test przeprowadzony został z wykorzystaniem środowiska UNICORE i pluginu EAOPT.

Geometria tarczy jest przedstawiona na rys. 10. Tarcza wykonana jest z materiału sprężysto-plastycznego z umocnieniem. Położenie pustki jest poszukiwane na podstawie zmierzonych przemieszczeń punktów pomiarowych tarczy przedstawionych na rys.10. Były poszukiwane trzy parametry kołowej pustki– średnica oraz współrzędne jej środka.



Rys. 10. Geometria tarczy i rozkład punktów pomiaru przemieszczeń

Fig. 10. The plates geometry and sensor points distribution

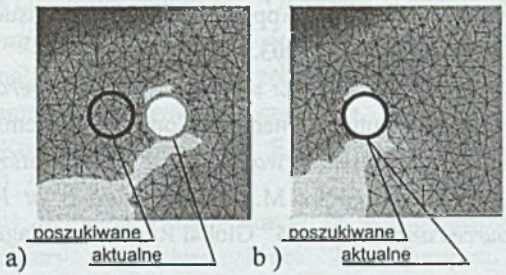
Funkcja przystosowania została wyrażona w postaci:

$$F = \sum_{i=1}^{n} |u_i - a_i|, \tag{4}$$

gdzie: n - liczba punktów pomarowych, u_i - przemieszczenie zmierzone w punkcie pomiarowym i, a_i - przemieszczenie obliczone w punkcie pomiarowym i dla aktualnego chromosomu z użyciem MES.

Parametry algorytmu ewolucyjnego: liczba podpopulacji - 2, liczba chromosomów - 10, liczba genów - 3. Użyte zostały operatory krzyżowania prostego I mutacji z rozkładem Ggaussa.

Najlepsze rozwiązanie uzyskane w pierwszym i ostatnim kroku algorytmu ewolucyjnego pokazano na rys.11. Odcieniami szarości zaznaczono wartości naprężeń redukowanych, jasny kolor określa strefy plastyczne.



Rys. 11. Wyniki identyfikacji, najlepsze rozwiązanie w: a) pierwszej, b) ostatniej generacji
Fig. 11. The results of identification, the best result in: a) first, b) last generation

Wyniki optymalizacji w postaci liczbowej przedstawiono w tab. 1.

Tabela 1

Wyniki identyfikacji

Zmienna	Wartość poszukiwana	Wartość zidentyfikowana	Błąd [%]
Położenie x	35.000	35.001	0.005
Położenie y	50.000	48.980	2.040
Promień r	10.000	10/.064	0.646

6. Wnioski

Zastosowanie środowisk gridowych stwarza nowe możliwości rozwoju metod optymalizacji i identyfikacji ewolucyjnej układów mechanicznych. Poprzez zastosowanie rozwiązań gridowych upraszcza się proces wykorzystania zaawansowanych zasobów

sprzętowych oraz programowych. Opracowanie metod oraz programów współpracujących ze środowiskami gridowymi pozwoli w niedługim czasie rozwiązywać bardzo złożone zadania optymalizacji i identyfikacji z wykorzystaniem powstających w ramach europejskich i światowych projektów gridów obliczeniowych.

Praca finansowana z projektu badawczego KBN 4T11F00822.

Literatura

1. Alda W., Górecki R., Budyn R., Ciesielski M., Kitowski J., *Prototype system for distributed scientific visualisation*, 3rd Cracow Grid Workshop, 2003.
2. Allen G., Davis K., Dolkas K. N., Doulamis N. D., Goodale T., Kielmann1 T., Merzky A., Nabrzyski J., Pukacki J., Radke T., Russell M., Seidel E., Shalf J., Taylor I., *Enabling Applications on the Grid*: A GridLab Overview, International Journal of High Performance Computing Applications: Special issue on Grid Computing: Infrastructure and Applications, 2003.
3. Almond J., Snelling D., *UNICORE: uniform access to supercomputing as an element of electronic commerce*, Future Generation Computer Systems,vol. 15, 1999.
4. Arabas J., *Wykłady z algorytmów ewolucyjnych*. WNT, Warszawa, 2001.
5. Bała P., Lesyng B., Niezgódka M., *GRID Solutions at ICM: a Contribution to European resources and services*, 4th Global Research Village Conference, Warszawa 2002.
6. Strony internetowe projektu BioGRID, http://biogrid.icm.edu.pl/.
7. Bubak M., Malawski M., Zając K., *Grid architecture for interactive applications*, Lecture Notes on Computational Science 3019, pp. 812-821, 2004.
8. Burczynski T., *Metoda elementów brzegowych w mechanice*, WNT, Warszawa 1995.
9. Burczyński T., Orantek P., *Hybrid evolutionary algorithms aided by sensitivity information in identification and structural optimization*. Proc. 2nd European Conference on Computational Mechanics, ECCM-2001, CD, Kraków, CD, 2001.
10. Burczyński T., Beluch W., Długosz A., Orantek P., Nowakowski M., *Evolutionary methods in inverse problems of engineering mechanics*. In: Inverse Problems in Engineering Mechanics II (eds. M.Tanaka and G.S.Dulikravich), Elsevier, 2000, 553-562.
11. Burczyński T., Beluch W., Długosz A., Nowakowski M., Orantek P., *Coupling of the boundary element method and evolutionary algorithms in optimization and identification problems*. ECCOMAS 2000, CD, Barcelona 2000.
12. Burczyński T., Orantek P., *Evolutionary algorithms aided by sensitivity information*. In: Artificial Neural Nets and Genetic Algorithms (eds. V.Kurkova at al.), Springer Verlag, Berlin 2001, pp.272-275.

13. Burczyński T., Długosz A., *Application of boundary element method in identification problems of thermoelasticity.* In: Advances in Boundary Element Techniques II (eds. M.Denda, M.H.Aliabadi, A.Charafi), Geneva, Hoggar 2001, 563-570.
14. Burczyński T., Beluch W., Długosz A., Kuś W., Orantek P., *The finite and boundary elements in evolutionary optimization.* Proc. M^2E'2000, Gliwice 2000, 53-60.
15. Burczyński T., Orantek P., *Połączenie algorytmów genetycznych i gradientowych.* Mat.III Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Złoty Potok 1999, 47-54.
16. Burczyński T., Beluch W., *Evolutionary identification and optimization of boundary conditions for cracked structures.* Mat.V Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra 2001.
17. Burczyński T., Kuś W., *Application of the distributed evolutionary algorithms in the shape optimization of elasto-plastic structures.* Mat.V Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra 2001.
18. Burczyński T., Kuś O., Nowakowski M., Orantek P., *Evolutionary algorithm in nondestructive identification of internal defects.* Mat.V Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra 2001.
19. Burczyński T., Orantek P. *Evolutionary approach to topology optimization of structures.* Mat.V Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra 2001, 56-63.
20. Burczyński T., Kuś W., *Shape optimization of elasto-plastic structures using distributed evolutionary algorithms.* Proc. 2nd European Conference on Computational Mechanics, ECCM 2001, Kraków 2001.
21. Burczyński T., Beluch W., Długosz A., Kokot G., Kuś W., Orantek P. *Evolutionary BEM computation in shape optimization problems.* IUTAM/IACM/IABEM Symposium on Advanced Mathematical and Computational Mechanics Aspects of the Boundary Element Method, ed. Burczynski T., Kluwer 2001.
22. Burczyński T., Kuś W., *Zastosowanie algorytmów genetycznych w optymalnym kształtowaniu ciał sprężysto-plastycznych.* Informatyka w technologii materiałów, Nr 3-4, pp. 189-195, Kraków, 2001.
23. Burczyński T., Kuś W. *Shape optimization of elasto-plastic structures using coupled BEM-FEM approach and distributed evolutionary algorithm.* Proc. World Conference on Computational Mechanics, WCCM 2002, Wien 2002.
24. Burczyński T., Kuś W., *Distributed evolutionary algorithms in shape optimization of nonlinear structures.* Lectures Notes on Computational Science 2328, Springer Verlag, Berlin 2002.
25. Burczyński T., Kuś W., *Boundary elements and distributed evolutionary algorithms in optimization of elasto-plastic structures.* Boundary Element Techniques (ed. Z. Yao, M.H. Aliabadi), pp. 151-156, Springer Verlag, Berlin 2002

26. Burczyński T., Kuś W., *Distributed evolutionary algorithm - tests and applications*, AI-METH 2002, Gliwice, 2002.
27. Burczyński T., Beluch W., Kuś W., Nowakowski M., Orantek P., *Evolutionary algorithms in selected optimization and identification problems of mechanical systems*. Chapter in: Evolutionary Computation and Global Optimization (ed. J. Arabas). Oficyna Politechniki Warszawskiej, Warszawa 2002.
28. Burczyński T., Kuś W., *Distributed and parallel evolutionary algorithms in optimization of nonlinear structures*. Proc. 15th International Conference on Computer Methods in Mechanics CMM-2003, Wisła 2003.
29. Burczyński T., Kuś W., Długosz A., *Distributed evolutionary algorithms in optimization of thermoelastic structures*. Proc. 15th International Conference on Computer Methods in Mechanics CMM-2003, Wisła 2003.
30. Burczyński T., Kuś W., Orantek P., *Distributed evolutionary algorithms in selected identification problems*. Proc. 15th International Conference on Computer Methods in Mechanics CMM-2003, Wisła 2003.
31. Burczyński T., Kuś W., Orantek P., Dlugosz A., *Distributed evolutionary algorithms in structural optimization and defect identification*. In: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems EUROGEN 2003, Barcelona 2003.
32. Burczyński T., Kuś W., *Distributed evolutionary algorithms with master-co-evolutionary algorithm and slaves - fitness function evaluators*. Prac. VI Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna KAEiOG 2003, 26-28 maja, Łagów 2003.
33. Burczyński T., Kuś W., *Improvements of distributed evolutionary algorithm in optimization of nonlinear structures*. Symposium on Methods of Artificial Intelligence AI-METH 2003, Gliwice 2003.
34. Burczyński T., Kuś W., *Optymalne projektowanie konstrukcji sprężysto-plastycznych z wykorzystaniem sekwencyjnych i rozproszonych algorytmów ewolucyjnych*. Rozdział 3 w: Informatyka w technologii materiałów, Informatyka w technologii metali, (eds. A..Piela, F.Grosman, J.Kusiak, M.Pietrzyk), Gliwice, pp.108-142, 2003.
35. Burczyński T., Kuś W., *Distributed evolutionary algorithms in optimization of nonlinear solids*. In: Evolutionary Methods in Mechanics (eds. T.Burczyński and A.Osyczka).. Kluwer 2004.
36. Kuś W., Burczyński T., *Computer implementation of the coevolutionary algorithm with Condor scheduler*, Mat. KAEIOG 2004, Kazimierz2004, pp. 109-114.
37. Burczyński T., Kuś W., Długosz A., Orantek P., *Optimization and defect identification using distributed evolutionary algorithms*, Engineering Applications of Artificial Intelligence, No.4, Vol. 17, 2004, 337-344.

38. Burczyński T., Kuś W., Długosz A., Poteralski A., Szczepanik M., *Sequential and distributed evolutionary computations in structural optimization.*, Lecture Notes on Artificial Intelligence 3070, Springer, 2004
39. Strony projektu CLUSTERIX: National CLUSTER of Linux systems http://www.clusterix.pcz.pl
40. Condor Project Homepage, http://www.cs.wisc.edu/condor/
41. Strony projektu CROSSGRID (IST-2001-32243) http://www.crossgrid.org
42. Długosz A., Burczyński T., *Evolutionary optimization in thermoelastic problems using the boundary element method.* In: Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2000, pp.145-150.
43. Strony projektu EGEE Enabling Grids for E-sciencs in Europe, http://public.eu-egee.org/
44. Strony internetowe projektu Eurogrid: *Application Testbed for European GRID computing*, IST 1999-20247 http://www.eurogrid.org/
45. Ferreira L. i inni, *Introduction to grid computing with Globus*, IBM Redbook, ibm.com/redbooks, 2003.
46. Foster I., Kesselman C., Tuecke S., *The anatomy of the grid, enabling scalable virtual organizations*, Int. J. High Perform. Comput. Appl. Vol. 15(3), 2001.
47. Foster I., Kesselman C., *Computational Grids*, chapter 2, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kauffman Publishers, 1999.
48. Strony internetowe Global Grid Forum, http://www.ggf.org/
49. Strony internetowe The Globus Alliance, http://www.globus.org/
50. Goldberg D.E., *Algorytmy genetyczne i ich zastosowania.* WNT, Warszawa 1995.
51. Graham P., Heikkurien M., Nabrzyski J., Oleksiak A., Parsons M., Stockinger H., Stockinger K., Stroiski M., Węglarz J.,*EU funded grid developement in Europe*, 2nd European Across Grids Conference, Cyprus 2004.
52. Strony projektu GRIDLAB (IST-2001-32133) http://www.gridlab.org
53. Strony internetowe projektu GRIP IST-2001-32257, http://www.grid-interoperability.org/
54. Johnston W. E., *Computational and Data Grids in large-scale science and engineering*, Future Genartion Computer Systems, vol. 18, 2002.
55. Kleiber M., Węglarz J.: *Polski Internet Optyczny jako wkład nauki w budowę społeczeństwa informacyjnego*, Polski Internet Optyczny: Technologie, Usługi i Aplikacje - PIONIER 2002, Poznań 2002.
56. Kleiber M.(ed), *Handbook of Computational Solid Mechanics*, Springer-Verlag, Berlin 1998.
57. Kozakiewicz A., Karbowski A., Błaszczyk J., Kubica B., Karpowicz M., *Computing grids: technology overview*, Mat. KAEIOG 2004, Kazimierz, pp. 251-261, 2004.
58. Kuczyński T., Wyrzykowski R., Żola J., *Web Access to Distributed Condor Pools*, 3rd Cracow Grid Workshop, 2003.

59. Kuś W., Burczyński T. *Distributed evolutionary algorithms in optimization of elasto-plastic structures with use of coupled FEM-BEM method.* Proc. AI-MECH 2001 Symposium on Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2001.
60. Kokot G., *Generlaized shape optimization using genetic algorithms and boundary elements.* In: Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2000, pp.197-202.
61. Kuś W., Burczyński T., *Evolutionary optimization of elasto-plastic solids.* In: Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2000, pp. 213-218.
62. Lesyng B., Bała P., Erwin D., *EUROGRID – European grid testbed*, Journal of Parallel and Distributed Computing, vol. 63, 2003.
63. Michalewicz Z., *Algorytmy genetyczne + struktury danych = programy ewolucyjne.* WNT, Warszawa 1996.
64. Strony internetowe projektu Nimrod-G, http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm
65. Strony projektu Ninf project http://ninf.apgrid.org/
66. Orantek P., Burczyński T., *Hybrid evolutionary algorithms aided by sensitivity information in identification and structural optimization.* In: Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2000, pp. 247-254.
67. Strony projektu PROGRESS http://progress.psnc.pl
68. Romberg M., *The UNICORE Grid infrastructure*, Scientific Programming, vol. 10 Issue 2, 2002.
69. Schaefer R., *Podstawy genetycznej optymalizacji globalnej*, Wyd. Uniw. Jagielońskiego, Kraków, 2002.
70. Strony projektu: Obliczenia wielkiej skali i wizualizacja do zastosowań w wirtualnym laboratorium z użyciem klastra SGI http://www.wcss.wroc.pl/pb/sgigrid/
71. Szczepanik M., Kuś W., Burczyński T., *Evolutionary optimization of plate systems.* In: Methods of Artificial Intelligence in Mechanics and Mechanical Engineering (eds. T.Burczyński and W.Cholewa), Gliwice 2000.
72. Strony projektu Teragrid http://www.teragrid.org/
73. Strony internetowe projektu UNICORE Forum, http://www.unicore.org/
74. Strony internetowe projektu UNICORE http://unicore.sourceforge.net/
75. Wyrzykowski R., Meyer N., Stroinski M., CLUSTERIX: *National cluster of linux systems.* 2nd European Across Grids Conference, Nicosia, Cyprus, Jan. 28-30, 2004.

Abstract

The shape optimization problem of structures can be solved using methods based on sensitivity analysis information or non-gradient methods based on genetic algorithms. The applications of evolutionary algorithms in optimization need only information about the values of an objective (fitness) function. The use of grid techniques in optimizations can lead to improvements in hardware and software utilization. The other advantages of grids are simple and uniform end user communication portals/programs. The idea presented in the paper is to prepare a group of plugins and programs for evolutionary optimization of structures using UNICORE environment.