OBLICZENIA RÓWNOLEGŁE W OPTYMALIZACJI LINIOWEJ I NIELINIOWEJ. ALGORYTM PULSACYJNY W OPTYMALIZACJI NIELINIOWEJ.

J.Sobczyk A.P.Wierzbicki *

29 czerwca 1995

Streszczenie

Współczesne komputery równoległe składające się z wielu procesorów o dużej mocy obliczeniowej skłaniają do podjęcia badań nad nowymi algorytmami pozwalającymi wykorzystać ich możliwości. Zrównoleglanie algorytmów optymalizacji jest zazwyczaj motywowane dużymi rozmiarami rozwiązywanych zadań. Duża złożoność obliczeniowa zadań optymalizacji nie musi być wynikiem dużych rozmiarów, może wynikać z innych przyczyn takich jak np. złe uwarunkowanie numeryczne. Wobec tego istnieje również potrzeba rozwoju algorytmów rozwiązujących zadania o średnim rozmiarze ale trudne z innych przyczyn. Jednym z podejść w tej dziedzinie może być gruboziarniste zrównoleglenie obliczeń pozwalające na jednoczesne sprawdzanie wielu alternatywnych kierunków poprawy. W taki rozwiązaniu każdy procesor wykonuje niezależnie pewien fragment algorytmu optymalizacji. Pozwala to zmniejszyć niezbędne narzuty na transmisję danych i koordynację obliczeń. Artykuł ten prezentuje klasę takich algorytmów dla optymalizacji nieliniowej bez ograniczeń, zwanych algorytmami pulsacyjnymi gdyż w kolejnych fazach następuje na przemian zwiększenie lub zmniejszenie rozrzutu uzyskiwanych rozwiązań. Przykładami takich algorytmów mogą być zmodyfikowany, równoległy algorytm typu Newtona lub równoległy algorytm zmiennej metryki rzędu pierwszego. W tym ostatnim przypadku zrównoleglenie pozwala na uniknięcie znanych wad zmiennej metryki rzędu pierwszego. Algorytm równoległy pozwala nie tylko przyspieszyć obliczenia ale również zwiększyć odporność na złe uwarunkowanie numeryczne.

1 Algorytm pulsacyjny - nowa metoda zrównoleglania zadań optymalizacji

W chwili obecnej istnieje wiele rozmaitych podejść do problemu zrównoleglania obliczeń w metodach optymalizacji, patrz np. [Bertsekas et al. (1989)], [Grauer et al. (1991)]. Wykorzystana tu metoda zrównoleglania obliczeń była opisana w [Wierzbicki (1993)]. To podejście ma na celu nie tyle przyspieszenie obliczeń dla zadań o dużych rozmiarach co wykorzystanie dostępnej mocy obliczeniowej komputerów równoległych do zwiększenia szybkości i niezawodności rozwiązywania zadań trudnych do rozwiązania innymi metodami, np. źle uwarunkowanych numerycznie. Podejście to charakteryzuje się następującymi cechami:

1. Algorytm optymalizacji (ale nie koniecznie samo zadanie optymalizacji - patrz [Ruszczyński (1989)]) jest dekomponowany na zadania, które mogą być wykonywane równolegle z różnymi wartościami parametrów.

*Instytut Automatyki i Informatyki Stosowanej, Politechnika Warszawska, ul. Nowowiejska 15/19, 00-665 Warszawa.

2. Rezultaty uzyskane z każdego takiego zadania są, wraz z wartościami użytych parametrów, wykorzystywane do sterowania całym procesem optymalizacji, a dzięki temu przyspieszenia i zwiększenia odporności algorytmu.

Oczywiście te cechy są bardzo ogólne i ich zastosowanie wymaga umiejętnego doboru szczegółowych rozwiązań. Rozwiązania te obejmują:

A Wybór ziarnistości zrównoleglenia. W tym przypadku koncentrujemy się na zrównolegleniu gruboziarnistym. Wiele dotychczasowych metod zrównoleglania dążyło do równoległości drobnoziarnistej np. zrównoleglania operacji macierzowych - patrz [Nogi (1986)]. Jednak nawet w takich przypadkach zrównoleglenie nie powinno być zbyt drobnoziarniste aby nie utracić efektywności wynikającej z przetwarzania potokowego w operacjach wektorowych - patrz [Carey (1989)]. Rozwój metod zrównoleglania drobnoziarnistego związany był z faktem, że większość dotychczasowych konstrukcji maszyn równoległych zawierała procesory wektorowe lub duże ilości procesorów o małej mocy obliczeniowej (komputery masywnie równoległe). Jednakże w dzisiejszych czasach coraz więcej komputerów równoległych oferuje architektury zawierające dziesiątki a nawet setki procesorów o mocy obliczeniowej porównywalnej z dawnymi superkomputerami. Stąd też skupienie się na metodach o równoległości gruboziarnistej wydaje się uzasadnione.

Równoległość gruboziarnista narzuca inne spojrzenie na proces obliczeniowy. W tym przypadku problemy związane z ilością transmitowanych danych mogą stać się mało istotne, natomiast większego znaczenia nabierają problemy związane z koordynacją obliczeń i sterowaniem nimi. W szczególności pojawia się możliwość asynchronicznego generowania i rozwiązywania poszczególnych podproblemów i adaptacyjnego sterowania obliczeniami.

B Wybór celu zrównoleglania. Potrzeba rozwiązywania dużych zadań optymalizacji jest wystarczającym uzasadnieniem do zrównoleglania obliczeń. Jednakże potrzeba taka może również wystąpić przy zadaniach mniejszych ale trudnych do rozwiązania z innych niż rozmiar powodów. W takich przypadkach możemy próbować równolegle rozwiązywać różne warianty tego samego zadania dla różnych wartości parametrów takich jak: punkt startowy, współczynniki skalujące, kryteria stopu itp. Takie podejście w oczywisty sposób zwiększa nakład obliczeń ale jednocześnie zwiększa niezawodność metody i w rezultacie często przyspiesza uzyskanie rozwiązania. Cała trudność polega na właściwym doborze proporcji pomiędzy dekompozycją i dywersyfikacją zadań, gdyż w różny sposób wpływa on na odporność i przyspieszenie metody.

Załóżmy, że mamy do dyspozycji P równolegle pracujących procesorów. Jeśli wygenerujemy P różnych wariantów parametrycznych danego problemu to uzyskamy wzrost odporności za cenę P-krotnego wzrostu nakładów obliczeniowych. Jeśli zaś dokonamy dekompozycji to rezultaty częściowe mogą być użyte zarówno do przyspieszenia obliczeń jak i zwiększenia odporności metody. W tym przypadku całkowity nakład obliczeń również będzie większy od nakładu obliczeń niezbędnego do rozwiązania szeregowego (choćby ze względu na komunikację międzyprocesorową i koordynację zadań). Dlatego też nie należy oczekiwać przyspieszenia proporcjonalnego do liczby użytych procesorów. Wydaje się, że rozsądnym może być oczekiwanie przyspieszenia rzędu $\sqrt{P}$ pod warunkiem, że pozostałe $\sqrt{P}$ zwiększonego nakładu obliczeń nie będzie całkowicie zmarnowane ale użyte do zwiększenia odporności metody.

Postulat '$\sqrt{P}$' jest oczywiście tylko przybliżonym celem. Generalnie ilość wariantów parametrycznych może być różna (oznaczmy ją M) od ilości procesorów. Liczbę wariantów można dobierać dynamicznie w zależności od rozmiaru problemu, liczby dostępnych procesorów a także dotychczas uzyskanych rezultatów. W zadaniach linowych dużej skali rozmiar zadania wyrażony liczbą kolumn i wierszy może sięgnąć setek tysięcy. W takim przypadku problemem może być ograniczenie się do liczby dostępnych procesorów i ich wielokrotne używanie do rozwiązywania kolejnych podproblemów. W naszym przypadku koncentrujemy się na zadaniach o średnim rozmiarze, których trudność wynika z innych przyczyn niż rozmiar. W szczególności, problemy optymalizacji nieliniowej mogą być trudne nawet jeśli ich rozmiary nie przekraczają kilkuset.

Problemy tego rodzaju o silnych nieliniowościach i złożonej strukturze mają liczne zastosowania w nauce i technice. W tym artykule koncentrujemy się na sposobie zastosowania ogólnej metody zrównoleglania do zadań nieliniowych, rozpoczynając od podstawowych algorytmów dla zadań bez ograniczeń i funkcji różniczkowalnych, gdyż są one również częścią składową algorytmów rozwiązujących zadania z ograniczeniami i nieróżniczkowalne. W tej dziedzinie trzy klasy algorytmów są uważane za podstawowe:

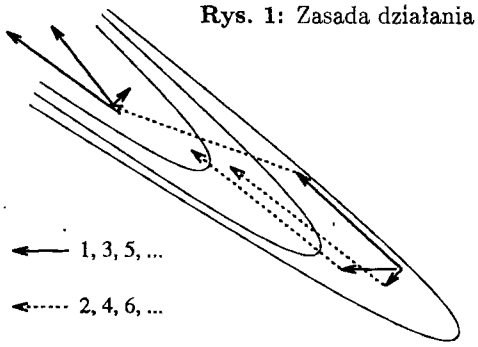
1. Różne odmiany metody kierunków sprzężonych. Algorytmy tego rodzaju są często używane również do rozwiązywania dużych układów równań liniowych. W tym przypadku były one zrównoleglane drobnoziarniście na różne sposoby - patrz [Bertsekas et al. (1989)]. Do stworzenia w pełni równoległej, gruboziarnistej metody konieczne jest znalezienie sposobu generowania zbiorów kierunków sprzężonych w jednym kroku. W ostatnim czasie pojawiło się kilka prób znalezienia takich metod - patrz [Chronopulos et al. (1989)], ale te próby są raczej związane z układami liniowymi lub zadaniami kwadratowymi. Natomiast pytanie czy uda się gruboziarniście zrównoleglić tę klasę algorytmów nadal pozostaje otwarte.
2. Różne warianty metod zmiennej metryki (quasi Newtonowskie). Metody te wykorzystują aproksymację hesjanu (lub jego odwrotności) tworzoną na podstawie przyrostów gradientu. Najpopularniejsze metody tego rodzaju (w tym BFGS i inne metody rzędu drugiego) są związane z ideą kierunków sprzężonych i w związku z tym, mogą sprawiać kłopoty w zrównolegleniu. Jednakże jednym z głównych rezultatów niniejszej pracy jest pokazanie, że inny wariant metody zmiennej metryki - metoda rzędu pierwszego może być dzięki jej specyficznym właściwiściom łatwo zrównoleglona, a jej słabości pokonane dzięki implementacji równoległej.
3. Różne warianty metod Newtonowskich korzystające z bezpośredniego obliczania drugich pochodnych. Należy zauważyć, że w ostatnich czasach obserwuje się nawrót do metod tego rodzaju, prawdopodobnie związany z rozwojem metod różniczkowania symbolicznego pozwalającego wyeliminować błędy w programowaniu pochodnych (np. dla problemu o dziesięciu zmiennych trzeba zaprogramować 55 drugich pochodnych). Zakładając, że odpowiednie narzędzia różniczkowania symbolicznego będą częścią oprogramowania optymalizacyjnego - patrz [Paczyński (1993)], niektóre z metod tego rodzaju mogą być wyjątkowo użyteczne.

Podstawowym elementem metod optymalizacji nieliniowej bez ograniczeń jest zazwyczaj optymalizacja kierunkowa. Różne metody optymalizacji nieliniowej różnią się głównie sposobem generowania kierunków poprawy. Z tego powodu optymalizacja kierunkowa będzie podstawowym zadaniem liczonym równolegle. Ponadto zakładamy, że optymalizacje kierunkowe są zgrupowane w specjalny dwufazowy równoległy algorytm pulsacyjny.

Załóżmy, że generujemy M wariantów parametrycznych. W najprostszym przypadku można je wygenerować jako poszukiwania w kierunkach wersorów. W bardziej złożonych przypadkach mogą one odpowiadać różnym aproksymacjom lub regularyzacjom hesjanu lub też różnym kierunkom sprzężonym albo różnym wariantom poprawek w programowaniu linowym, itp. W nieparzystych (rozbieżnych) iteracjach algorytmu pulsacyjnego optymalizacje kierunkowe są przeprowadzane równolegle w tych kierunkach i w rezultacie otrzymujemy rozproszony zbiór rozwiązań aproksymujących rozwiązanie optymalne (patrz – rys 1). W następnej iteracji z każdego z tych punktów wytyczamy nowy kierunek poprawy i dokonujemy optymalizacji kierunkowej. Te nowe kierunki powinny (przynajmniej w dostatecznie małym otoczeniu rozwiązania optymalnego) być zbieżne i prowadzić do dobrej aproksymacji optimum. W dużej odległości od punktu optymalnego nawet te zbieżne aproksymacje mogą być dość znacznie rozrzucone ale ten rozrzut pozwala na poprawienie odporności algorytmu na błędy numeryczne, złe uwarunkowanie zadania a nawet do pewnego stopnia na minima lokalne. Taka podwójna iteracja kończy się wyborem najlepszego ze znalezionych rozwiązań, które stanie się punktem startowym dla następnych iteracji.

Zasada działania algorytmu pulsacyjnego jest jak widać bardzo ogólna i może być zastosowana w szerokiej klasie algorytmów różnych typów: z ograniczeniami, nieróżniczkowalnych

Rys. 1: Zasada działania algorytmu pulsacyjnego



itp. Tutaj jednak skoncentrujemy się na przypadku zadań różniczkowalnych bez ograniczeń i metodach Newtonowskich lub quasi-Newtonowskich.

2 Równoległe warianty metod Netonowskich lub quasi-Newtonowskich

Załóżmy, że problem jest nieliniowy bez ograniczeń a funkcja celu $f : \boldsymbol{R}^n \to \boldsymbol{R}^1$ dwukrotnie różniczkowalna. Zmienna metryka jest to macierz $n \times n$ oznaczana $\boldsymbol{V}^{(k)}$ gdzie k jest numerem iteracji. Macierz ta jest aproksymacją hesjanu funkcji celu $\boldsymbol{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$, lub - jak to jest przyjęte tutaj - aproksymacją odwrotności hesjanu $\boldsymbol{H}^{-1}(\boldsymbol{x})$. We wstępnej analizie takiej aproksymacji zakłada się, że funkcja minimalizowana jest kwadratowa:

$$\begin{aligned} f(\boldsymbol{x}) &= 0.5 < \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x} > + < \boldsymbol{b}, \boldsymbol{x} > + c \\ \nabla f(\boldsymbol{x}) &= \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{g} \in \boldsymbol{R}^n \\ \nabla^2 f(\boldsymbol{x}) &= \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{A} \in L(\boldsymbol{R}^n, \boldsymbol{R}^n) \end{aligned} \tag{1}$$

gdzie $< ., . >$ oznacza iloczyn wewnętrzny. W takim przypadku hesjan jest stały. W metodzie iteracyjnej przyrosty gradientu odpowiadające przyrostom $\boldsymbol{s}^{(k)}$ zmiennej $\boldsymbol{x}$ w kolejnych krokach oznaczane są przez $\boldsymbol{y}^{(k)}$, Przy takich oznaczeniach dla funkcji kwadratowej o dodatnio określonym hesjanie zachodzą następujące zależności:

$$\begin{aligned} \boldsymbol{s}^{(k)} &= \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \\ \boldsymbol{y}^{(k)} &= \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)}) \\ \boldsymbol{y}^{(k)} &= \boldsymbol{H}\boldsymbol{s}^{(k)} \\ \boldsymbol{s}^{(k)} &= \boldsymbol{H}^{-1}\boldsymbol{y}^{(k)} \end{aligned} \tag{2}$$

Wobec tego można utworzyć macierz $\boldsymbol{Y}$ przyrostów gradientu $\boldsymbol{y}^{(k)}$ odpowiadających (n liniowo niezależnym) krokom $\boldsymbol{s}^{(k)}$ tworzącym macierz $\boldsymbol{S}$ dla $k = 1,..n$ i obliczyć $\boldsymbol{H}^{-1} = \boldsymbol{S}\boldsymbol{R}^{-1}$. Jednakże często bardziej użyteczne bywa zastosowanie aproksymacji przyrostowej $\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \Delta\boldsymbol{V}^{(k)}$ odwrotności hesjanu $\boldsymbol{H}^{-1}$ skonstruowanej tak by zachodziły następujące własności:

$$\boldsymbol{s}^{(j)} = \boldsymbol{V}^{(k+1)}\boldsymbol{y}^{(j)} \qquad \text{for} \quad j = 1,..k \tag{3}$$

co implikuje, że $\boldsymbol{V}^{(n+1)} = \boldsymbol{H}^{-1}$, jeśli wszystkie $\boldsymbol{y}^{(k)}$ są liniowo niezależne.

Znanych jest wiele formuł na $\Delta V^{(k)}$ które — przy odpowiednich założeniach — spełniają powyższe wymagania. Najpopularniejsze są metryki rzędu drugiego takie jak BFGS lub jej metoda dualna DFP – patrz [Fletcher (1987)], [Gill et al. (1981)]. Niestety założenie (3) dla zmiennych metryk rzędu drugiego zachodzi jedynie pod warunkiem dokładnej optymalizacji kierunkowej w następujących quasi–Newtonowskich kierunkach:

$$\begin{aligned} d^{(k)} &= -V^{(k)} g^{(k)} \text{ for } k > 1 \\ d^{(1)} &= -g^{(1)} \\ g^{(k)} &= \nabla f(x^{(k)}) \end{aligned} \tag{4}$$

i jest związane z faktem, że te kierunki są sprzężone. Z drugiej strony udowodniono [Stachurski (1981)], że w klasie metod zmiennej metryki rzędu drugiego zwanej klasą metod Broydenowskich (zawierająca metody DFP i BFGS) warunek (3) jest spełniany asymptotycznie dla funkcji dwukrotnie różniczkowalnych z dodatnio określonym i ograniczonym hesjanem. Rezultatem jest superliniowa zbieżność nawet przy braku precyzyjnej optymalizacji kierunkowej i sprzężenia kierunków, jeśli kroki quasi-Newtowskie $s^{(k)} = d^{(k)}$ są stosowane konsekwentnie. Widać więc, że ta właściwość zależy od kolejności obliczeń i w związku z tym zrównoleglenie metod zmiennej metryki rzędu drugiego może być trudne.

Na szczęście istnieje wariant metody zmiennej metryki nie wymagający kierunków quasi–Newtonowskich (4), precyzyjnej optymalizacji kierunkowej ani sprzężoności kierunków (4) do spełnienia warunku (3) i uzyskania aproksymacji odwrotności hesjanu. Jest on zdefiniowany następującą formułą:

$$\Delta V^{(k)} = \frac{(s^{(k)} - V^{(k)} y^{(k)}) \times (s^{(k)} - V^{(k)} y^{(k)})}{< (s^{(k)} - V^{(k)} y^{(k)}),\, y^{(k)} >} \tag{5}$$

gdzie . $\times$. oznacza iloczyn zewnętrzny. Metoda ta nie jest popularna gdyż jest obarczona zasadniczą wadą — staje się źle uwarunkowana gdy macierz $V^{(k)}$ dobrze aproksymuje H^{-1}. Wówczas mianownik (5) jest staje się bliski zera:

$$< (s^{(k)} - V^{(k)} y^{(k)}),\, y^{(k)} > = < (H^{-1} - V^{(k)}) y^{(k)},\, y^{(k)} > \tag{6}$$

Zabezpieczenie tej metody przed złym uwarunkowaniem jest możliwe i prowadzi do dobrych własności teoretycznych, takich jak monotoniczność aproksymacji H^{-1} oraz dobrej efektywności praktycznej – patrz [Kręglewski and Wierzbicki (1981)]. Dotychczas możliwości te raczej nie były wykorzystywane. Ostatnimi czasy podobne zabezpieczenia zostały zastosowane z dużym powodzeniem w metodzie zmiennej metryki rzędu pierwszego, która okazała się lepsza od metod rzędu drugiego – patrz [Conn et al. (1991)]. Co więcej, przy raczej słabych założeniach pokazano, że błędy aproksymacji hesjanu $\|H^{-1}(\hat{x}) - V^{(k)}\|$ mają ten sam rząd zbieżności co ciąg $\|x^{(k)} - \hat{x}\|$, gdzie $\hat{x}$ jest granicą $x^{(k)}$.

Dokładnie ta sama zmodyfikowana metoda zmiennej metryki rzędu pierwszego może być łatwo zrównoleglona. Ponieważ własność (3) nie zależy od dokładności optymalizacji kierunkowych ani sprzężoności kierunków, więc kroki $s^{(k)}$ i odpowiadające im przyrosty gradientu $y^{(k)}$ mogą być obliczane niezależnie. Jeden z procesorów może być zarezerwowany do obliczania aproksymacji (5) z odpowiednimi zabezpieczeniami, które są łatwiejsze i bezpieczniejsze w przypadku równoległym, gdyż nie musimy posiłkować się niekompletną aproksymacją $H^{-1}(x)$ i możemy rozpocząć nową iterację gdy pełna, dobrze uwarunkowana aproksymacja jest gotowa.

2.1 Podstawowy wariant metody zmiennej metryki.

Załóżmy, że rozmiar problemu n jest większy niż liczba dostępnych procesorów P. W tym przypadku, każdy procesor może być użyty do kilku zadań. Oznaczmy numer procesora przez

$\psi = 1, ...P$, gdzie procesor $\psi = 1$ będzie zarezerwowany do obliczania aproksymacji zmiennej metryki i koordynacji zadań. Do numerowania iteracji użyjemy indeksu K, który będzie wzrastał o dwa w każdej podwójnej iteracji.

W nieparzystych (rozbieżnych) iteracjach algorytmu pulsacyjnego, przypiszemy procesorom $\psi = 2, ...P$ następujące kierunki minimalizacji: pierwszemu $\boldsymbol{d}^{(1,0)} = -\frac{\boldsymbol{g}^{(1)}}{\|\boldsymbol{g}^{(1)}\|}$ lub $\boldsymbol{d}^{(K,0)} = -\boldsymbol{V}^{(K-1)}\boldsymbol{g}^{(K)}$ (największy spadek w pierwszej iteracji i quasi-Newtonowski w następnych), a pozostałym kierunki wersorów, $\boldsymbol{d}^{(K,j)} = \boldsymbol{e}_j = (0...1_{(j)}..0)^T$, $j = 1, ..n$. W przypadku szczególnym $P = n + 2$, każdy procesor z $\psi > 1$ będzie miał przypisany dokładnie jeden kierunek. Gdy procesorów jest mniej niż $n + 2$, co jest sytuacją typową, wówczas każdy procesor ma kolejkę χ kierunków.

Zasadniczym zadaniem procesora $\psi > 1$ jest wykonanie przybliżonej optymalizacji kierunkowej w każdym z tych kierunków rozpoczynając ze wspólnego dla wszystkich procesorów punktu $\boldsymbol{x}^{(K)}$.

$$\begin{aligned} \hat{\tau}^{(K,j)} &\approx \underset{\tau \in R^1}{\operatorname{argmin}} f(\boldsymbol{x}^{(K)} + \tau \boldsymbol{d}^{(j)}) \\ \boldsymbol{s}^{(K,j)} &= \hat{\tau}^{(K,j)} \boldsymbol{d}^{(j)} \qquad \text{for} \quad j = 0, \ldots n \end{aligned} \tag{7}$$

Są różne podejścia do optymalizacji kierunkowej. Jednym z częstszych wymagań jest żądanie dużej dokładności określenia minimum w danym kierunku w oparciu o aproksymację kwadratową, obliczanie gradientu przy obliczaniu wartości funkcji i test stopu Wolfa–Powella, patrz – [Fletcher (1987)].

Innym rozwiązaniem jest raczej zgrubna optymalizacja kierunkowa oparta na aproksymacji kwadratowej, unikająca obliczania gradientu i używająca testu stopy Goldsteina dopuszczającego jak najszybciej $\tau = 1$.

Wiadomo, że dokładna optymalizacja kierunkowa daje lepsze rezultaty praktyczne gdy jest używana z metodami kierunków sprzężonych lub zmiennej metryki rzędu drugiego.

Jednakże w tym algorytmie nie korzystamy z warunku sprzężoności kierunków nia ze zmiennej metryki rzędu drugiego. Dlatego też w iteracjach nieparzystych pulsara, zgrubna optymalizacja kierunkowa może być lepsza. W parzystych iteracjach gdzie konsekwentnie używane są kierunki quasi–Newtonowskie zgrubna optymalizacja kierunkowa również może być korzystna. Wobec tego opiszemy tu przypadek zgrubnej optymalizacji kierunkowej startującej z $\tau = 1$.

Zgrubna optymalizacja kierunkowa ma prawo zawieść. Dlatego, niech $\hat{\tau}^{(K,j)}$ oznacza wartość kroku przy której optymalizacja kierunkowa (7) się zatrzymała. Ta wartość kroku powinna spełniać test dwuskośny Goldsteina (patrz – [Fletcher (1987)]; my używamy w tym teście nierówności ostrych).

$$(1-\beta)\hat{\tau}^{(K,j)}\phi'^{(K,j)}(0) < \phi^{(K,j)}(\hat{\tau}^{(K,j)}) < \beta\hat{\tau}^{(K,j)}\phi'^{(K,j)}(0) \tag{8}$$

gdzie:

$$\begin{aligned} \phi^{(K,j)}(\tau) &= f(\boldsymbol{x}^{(K)} + \tau \boldsymbol{d}^{(j)}) - f(\boldsymbol{x}^{(K)}) \\ \phi'^{(K,j)}(0) &= < \nabla f(\boldsymbol{x}^{(K)}), \boldsymbol{d}^{(j)} > \\ \beta &\in (0; 0.5) \end{aligned}$$

Istnieją różne możliwości wprowadzania poprawek do zmiennej metryki. Jedną z możliwości jest:

$$\Delta \boldsymbol{V}^{(K,j)} = \begin{cases} \frac{\boldsymbol{r}_\alpha^{(K,j)} \times \boldsymbol{r}_\alpha^{(K,j)}}{<\boldsymbol{r}_\alpha^{(K,j)}, \boldsymbol{y}_\alpha^{(K,j)}>} & \text{if} \quad < \boldsymbol{r}_\alpha^{(K,j)}, \boldsymbol{y}_\alpha^{(K,j)} > \geq \gamma' \|\boldsymbol{y}_\alpha^{(K,j)}\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\begin{aligned} V^{(K,0)} &= \gamma'' I \\ V^{(K,j+1)} &= V^{(K,j)} + \Delta V^{(K,j)} \qquad \text{for} \quad j = 0, 1, \ldots n \end{aligned} \tag{10}$$

Poprawka (9) zmienia aproksymację odwrotności hesjanu tylko wtedy gdy dane wskazują, że wartości własne macierzy $H^{-1} - V^{(K,j)}$ są większe niż γ''. Współczynnik γ'' można interpretować jako ograniczenie tych wartości własnych. Jako alternatywy w tej poprawce można używać warunku: $< r_\alpha^{(K,j)} \ , \ y_\alpha^{(K,j)} > \geq \gamma'' \ \|y_\alpha^{(K,j)}\| \ \|s_\alpha^{(K,j)}\|$ W tym przypadku współczynnik γ'' można interpretować jako ograniczenie względnej dokładności $\frac{\|H^{-1} - V^{(K,j)}\|}{\|H^{-1}\|}$ w normie spektralnej.

Przy implementacji sekwencyjnej algorytmu zmiennej metryki, trzeba zachować szczególną ostrożność aby $s^{(k)}$ pozostały liniowo niezależne w kolejnych iteracjach. Uzyskuje się to przez wybór $d^{(K,j)} = e_j$ w algorytmie pulsacyjnym. W algorytmie szeregowym startowa macierz $V^{(0)}$ musi być dodatnio określona z niezbyt małymi wartościami własnymi np. $V^{(0)} = I$ ponieważ kierunek quasi-Newtonowski (4) musi być kierunkiem poprawy nawet jeśli aproksymacja odwrotności hesjanu nie jest kompletna. W algorytmie pulsacyjnym początkowa macierz $V^{(0)}$ nie jest niezbędna do wyznaczenia kierunku poprawy i dlatego można użyć $V^{K,0} = 0$. Proponujemy używanie jednego z dwu wariantów: $V^{K,0} = \gamma'' I$ gdzie $\gamma'' = \frac{1}{\mu}$ jest dolnym ograniczeniem wartości własnych hesjanu. To prowadzi do następującej własności aproksymacji:

$$0 < V^{(K,0)} \leq V^{(K,1)} \leq .. \leq V^{(K,j)} \leq V^{(K,j+1)} \leq .. \leq V^{(K,n)} \leq H^{-1} \tag{11}$$

gdzie nierówności są rozumiane w sensie dodatnio określoności różnic macierzy. Ten wariant nazywany jest aproksymacją od dołu.

Lub, jeśli mamy górne ograniczenie $\frac{1}{\nu}$ na wartości własne odwrotności hesianu (ν jest dolnym ograniczeniem wartości własnych hesjanu), możemy zastosować również aproksymację od góry przez użycie $\gamma'' = \frac{1}{\mu}$. W tym przypadku znaki nierówności (11) ulegną zmianie. Aproksymacja od góry daje zazwyczaj lepsze rezultaty numeryczne w algorytmie sekwencyjnym, patrz – citekreglewski. Jednakże aproksymacja od góry wymusza również zmianę znaku w warunku (9) na

$$\text{if} \quad < r_\alpha^{(K,j)}, y_\alpha^{(K,j)} > \leq -\gamma' \ \|y_\alpha^{(K,j)}\|^2$$

lub odpowiednio $\|y_\alpha^{(K,j)}\|^2$ musi być zastąpione przez $\|y_\alpha^{(K,j)}\| \ \|s_\alpha^{(K,j)}\|$.

3 Eksperymenty numeryczne.

Większości eksperymentów numerycznych została przeprowadzona na sieci stacji roboczych typu SUN z wykorzystaniem biblioteki "pvm"[1] Wcześniejsze eksperymenty wykonywane były na pojedynczej stacji SUN z symulacją równoległości (metodą współprogramów - ang. coroutines). A niektóre zostały wykonane na maszynie Paragon należącej do stowarzyszenia SINTEF w Trondheim w Norwegii.

Jako zadania testowania zostały wybrane problemy opisane w [Schittkowski (1987)]. A wśród nich następujące problemy:

Problem Sch_281

$$\begin{aligned} i &= 1..10 \\ f(x) &= [\sum_{i=1}^{10} i^3 (x_i - 1)^2]^{\frac{1}{3}} \end{aligned}$$

[1]Parallel Virtual Machine.

Problem Sch_286 Jest to łatwiejsze (pozbawione punktów stacjonarnych) uogólnienie funkcji Rosenbrocka.

$$\begin{aligned} i &= 1..20 \\ f(x) &= \sum_{i=1}^{10} 100(x_i^2 - x_{i+10})^2 + (x_i - 1)^2 \end{aligned}$$

Problem Sch_287

$$\begin{aligned} i &= 1..20 \\ f(x) &= \sum_{i=1}^{5} \{100(x_i^2 - x_{i+5})^2 + (x_i - 1)^2 + 90(x_{i+10}^2 + x_{i+15})^2 \\ &\quad +(x_{i+10} - 1)^2 + 10.1[(x_{i+5} - 1)^2 + (x_{i+15} - 1)^2] + 19.8(x_{i+5} - 1)(x_{i+15} - 1)\} \end{aligned}$$

Problem Sch_288

$$\begin{aligned} i &= 1..20 \\ f(x) &= \sum_{i=1}^{5} (x_i + 10x_{i+5})^2 + 5(x_{i+10} - x_{i+15})^2 + (x_{i+5} - 2x_{i+10})^4 + 10(x_i - x_{i+15})^4 \end{aligned}$$

Problem Sch_289

$$\begin{aligned} i &= 1..30 \\ f(x) &= 1 - exp[-\frac{1}{60}\sum_{i=1}^{30} x_i^2] \end{aligned}$$

Problemy Sch_290...293

$$f(x) = (x^T A x)^2$$

$$A = diag.(1, 2, ...N) = \begin{pmatrix} 1 & & & & & 0 \\ & 2 & & & & \\ & & 3 & & & \\ & & & \ddots & & \\ & & & & N-1 & \\ 0 & & & & & N \end{pmatrix}$$

$$f(x) = \left(\sum_{k=1}^{N} k x_k^2\right)^2$$

numer problemu	rozmiar N
Sch_290	2
Sch_291	10
Sch_292	30
Sch_293	50

Problemy Sch_294...299 Jest to trudniejsze uogólnienie funkcji Rosenbrocka.

$$f(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

numer problemu	rozmiar N
Sch_294	6
Sch_295	10
Sch_296	16
Sch_297	30
Sch_298	50
Sch_299	100

Problemy Sch_300...302

$$f(x) = x^T A x - 2x_1$$

$$A = \begin{pmatrix} 1 & -1 & & & & & & 0 \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & -1 & 2 & \ddots & & & \\ & & & -1 & \ddots & -1 & & \\ & & & & \ddots & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ 0 & & & & & & -1 & 2 \end{pmatrix}$$

numer problemu	rozmiar N
Sch_300s	4
Sch_300	20
Sch_301	50
Sch_302	100

Problemy Sch_303...305

$$f(x) = \sum_{i=1}^{N} x_i^2 + (\sum_{i=1}^{N} \frac{1}{2} i x_i)^2 + (\sum_{i=1}^{N} \frac{1}{2} i x_i)^4$$

numer problemu	rozmiar N
Sch_303s	10
Sch_303	20
Sch_304	50
Sch_305	100

Problemy Ros_8, Ros_16, Ros_32 Jest to trudniejsze uogólnienie funkcji Rosenbrocka.

$$f(x) = \sum_{i=0}^{N-1} [100(x_{2i+1} - x_{2i}^2)^2 + (1 - x_{2i})^2]$$

numer problemu	rozmiar N
Ros_8	8
Ros_16	16
Ros_32	32

4 Rezultaty

Po wykonaniu serii obliczeń na sieci zawierającej 8 stacji roboczych typu SUN uzyskano rezultaty przedstawione w tabelce. Wyjaśnienia wymaga tu kolumna opisana jako wirtualna ilość wywołań funkcji. Jest to suma maksymalnych ilości wywołań funkcji w każdej iteracji. Dość dobrze odzwierciedla więc ona faktyczną szybkość algorytmu. Jak widać z przedstawionych danych algorytm równoległy uzyskuje często znaczne przyspieszenia dla funkcji łatwych, natomiast jego szybkość dla funkcji trudnych jest porównywalna. Zasadniczą zaletą natomiast jest znacznie większa niezawodność algorytmu.

Problem	roz-miar	liczba itercji		liczba wywołań funkcji		
		m. równoległa	m. szeregowa	metoda równoległa		m. szeregowa
				całkowita	wirtualna	
Sch_300s	4	2	4	31	9	24
Sch_300	20	2	21	111	9	118
Sch_301	50	2	51	261	9	277
Sch_302	100	2	101	511	9	554
Sch_290	2	8	9	429	190	151
Sch_291	10	8	25	3546	370	506
Sch_292	30	32	66	19034	1051	1098
Sch_293	50	28	84	25233	702	1660
Sch_303s	10	4	3	234	58	61
Sch_303	20	6	6	1829	93	97
Sch_304	50	1	>300	126	126	>9893
Sch_289	30	1	1	98	98	38
Sch_288	20	18	39	6241	465	581
Sch_286	20	154	99	50497	3555	1857
Ros_8	8	38	55	7954	1201	1058
Ros_16	16	48	92	17539	1482	1507
Ros_32	32	60	139	50842	2278	2120
Sch_294	6	49	37	6130	1312	723
Sch_295	10	58	53	13831	1783	916
Sch_296	16	72	83	31004	2729	1553
Sch_297	30	108	146	74875	3466	2924
Sch_298	50	215	234	249489	6716	4618

Widać to było już z wcześniejszych obliczeń.

Rezultaty szeregowej i równoległej metody zmiennej metryki rzędu pierwszego.							
Iteration number	5	10	15	20	30	40	50
Szeregowa				zawiodła	zawiodła	zawiodła	zawiodła
norma rozwizania^2	0.68E+00	0.72E+00	0.71E+00				
norma gradientu^2	3.62E+01	3.34E+01	3.32E+01				
Pulsar							zawiódł
norma rozwizania^2	5.69E-04	1.76E-07	5.21E-11	4.46E-13	4.26E-17	1.48E-19	
norma gradientu^2	3.87E-09	1.03E-19	1.47E-24	1.54E-36	1.80E-48	1.07E-55	

Rezultaty te pokazują zasdniczy wzrost niezawodności jak również znaczące przyspieszenie.

Inny test dotyczył n-wymiarowego uogólnienia funkcji Rosenbrocka (tzw. funkcji bananowej). Rezultaty dotyczą łatwiejszego uogólnienia tej funkcji dla wymiarów i trudniejszego o wymiarze 20. Obliczenia zostały przeprowadzone na komputerze Paragon[2] zawierającym 56 procesorów i860.

[2]Paragon is a trademark of Intel Corporation.

Funkcja bananowa Rosenbrock'a – liczba iteracji niezbędna do uzyskania dokładności10^{-6} normy gradientu				
Rozmiar problemu	8	16	32	20 (wersja trudna)
Szeregowa	61	93	136	106
Równoległa (symulacja)	43	36	58	72
Równoległa (Paragon)	39	68	116	68

należy zauważyć, że przyspieszenie wynika nie tyle z pełnego wykorzystania mocy obliczeniowej co z własności metody, gdyż przy $\chi = 1$ większość procesorów jest niewykorzystana.

5 Wnioski.

Wstępne testy prostego wariantu algorytmu pulsacyjnego potwierdzają teoretyczne oczekiwania dotyczące możliwości tego algorytmu. Dwufazowy algorytm pulsacyjny okazuje się być bardziej odporny od algorytmu szeregowego. Dalsze prace powinny pozwolić na poprawienie przyspieszenia i zrównoleglenie innych algorytmów optymalizacji nieliniowej.

Literatura

[Bertsekas et al. (1989)] Bertsekas D.P. and J.N. Tsitsiklis (1989) *Parallel and Distributed Computation.* Prentice Hall, Englewood Cliffs.

[Carey (1989)] Carey, G.F., ed. (1989) *Parallel Supercomputing: Methods, Algorithms and Applications.* Wiley, Chichester - New York.

[Chronopulos et al. (1989)] Chronopoulsos, A.T., and C.W. Gear (1989) s-step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics* Vol. 25 pp. 153-168.

[Conn et al. (1991)] Conn, A.R., N.I.M. Gould and Ph.L. Toint (1991) Convergence of quasi-Newton matrices generated by the symmetric rank-one update. *Mathematical Programming* Vol. 50 pp. 177-195.

[Fletcher (1987)] Fletcher, R. (1987) *Practical Methods of Optimization.* Wiley, Chichester - New York.

[Gill et al. (1981)] Gill, E.P., W. Murray and M.H. Wright (1981) *Practical Optimization.* Academic Press, London-New York.

[Grauer et al. (1991)] Grauer, M. and D.B. Pressmar, eds. (1991) *Parallel Computing and Mathematical Optimization.* Springer-Verlag, Berlin-Heidelberg.

[Kręglewski and Wierzbicki (1981)] Kręglewski, T. and A.P.Wierzbicki (1981) Further properties and modifications of the rank-one variable metric method. In S. Walukiewicz and A.P. Wierzbicki, eds.: *Methods of Mathematical Programming*, PWN Warsaw.

[Nogi (1986)] Nogi, T. (1986) Parallel computation. In: Patterns and Waves - Qualitative Analysis of Nonlinear Differential Equations, *Studies in Mathematics and Applications* Vol. 18 pp. 279-318.

[Paczyński (1993)] Paczyński, J. (1993): Models of Dynamic Discrete Nonlinear Systems - Issues of Description Languages and Preprocessing. Report of the Institute of Automatic Control, Warsaw University of Technology.

[Ruszczyński (1989)] Ruszczyński, A. (1989) An augmented Lagrangian decomposition method for block diagonal linear programming problems, *Operations Research Letters 8*, pp. 287-294.

[Schittkowski (1987)] Schittkowski, K. (1987) *More test Examples for Nonlinear Programming Codes* Springer-Verlag, Berlin-Heidelberg.

[Stachurski (1981)] Stachurski, A. (1981) Superlinear convergence of Broyden's bounded θ-class of methods, *Mathematical Programming 20* pp. 196-212.

[Wierzbicki (1984)] Wierzbicki, A.P. (1984) Models and Sensitivity of Control Systems. Elsevier - WNT, Amsterdam-Warsaw.

[Wierzbicki (1993)] Wierzbicki, A.P. (1993) Augmented Simplex - a Modified and Parallel Version of Simplex Method Based on Multiple Criteria and Subdifferential Optimization Approach. Report of the Institute of Automatic Control, Warsaw University of Technology.