Wojciech BOŻEJKO, Mieczysław WODECKI
Politechnika Wrocławska
Piotr NADYBSKI
Wyższa Szkoła Zawodowa w Legnicy

PROBLEM REPLIKACJI DANYCH Z OGRANICZONĄ PRZEPUSTOWOŚCIĄ KANAŁÓW

Streszczenie. W pracy rozpatrujemy problem replikacji dużych zbiorów danych w chmurze obliczeniowej. Poprzez kanały o ograniczonej przepustowości rozsyłane są kopie do różnych lokalizacji. Każda z kopii ma ustalony rozmiar (czas transmisji), najpóźniejszy termin dostarczenia oraz współczynnik kary za jego przekroczenie. Należy tak zaplanować rozsyłanie, aby suma kar za przekroczenie terminów była minimalna. Przedstawimy model problemu, pewne własności, algorytmy przybliżone oraz wyniki przeprowadzonych eksperymentów obliczeniowych.

DATA REPLICATION PROBLEM WITH LIMITED CHANNEL CAPACITY

Summary. In the paper we consider the problem of replication of large data sets in the cloud computing. Through channels with limited bandwidth copies of various locations are sent. Each copy has a fixed size (transmission time), the latest delivery time and the penalty for exceeding it. One should schedule sending process such that the sum of penalties for deadlines exceeding should be minimal. We present a problem model, certain properties, approximate algorithms and results of computational experiments.

1. Wstęp

Malejący koszt usług związanych z przechowywaniem dużych zbiorów danych jest jednym z powodów rosnącego zainteresowania chmurami obliczeniowymi. Rozproszenie kopii danych oraz ich redundantność, daje łatwą dostępność i minimalizuje ryzyko utraty danych w przypadku awarii lub nieprzewidywalnych zdarzeń losowych. Decydując się na taki sposób przechowywania danych należy brać pod uwagę szereg uwarunkowań, od których zależy końcowa efektywność użytego rozwiązania. Wśród nich wymienić można ograniczenia związane z pojemnością zbiorów danych, przepustowością sieci, kosztami przechowywania oraz szeregiem innych, specyficznych dla danego modelu ograniczeń. Rozpatrywane w literaturze zagadnienia przechowywania oraz replikacji danych w dużej mierze są związane z problemami typowymi dla środowisk chmurowych. Zazwyczaj dotyczą metod optymalizacji kosztu przechowywania oraz minimalizacji ryzyka utraty danych. W pracach Mansouri i in. [7] oraz Thanasis i in. [9] jest rozpatrywane problem dostępności usług w architekturze wielochmurowej

w powiązaniu z kryterium kosztu. Z kolei Hajdi [5] oraz Liu i in. [6] uwzględniją dodatkowo problematykę związaną z przepustowością sieci, opóźnieniami oraz kosztami związanymi z przesyłaniem danych do miejsc ich docelowego składowania. Nieco inne podejście zostało przedstawione w pracy Yanzhena oraz Naixue [10], gdzie autorzy prezentują rozproszony algorytm wyboru docelowego miejsca wykonania kopii zapasowej niezależnie przez każdy z węzłów. Praca Djebbara i in. [4] dotyczy problemu zapewnienia dostatecznej jakości usług (akceptowalny poziom opóźnień, procent spóźnionych żądań, itp.) dostarczanych on-line. Autorzy przedstawiają model matematyczny oraz proponują różne metody poprawy efektywności, np. poprzez wprowadzenie priorytetów lub dynamiczne zarządzanie wybudzaniem "uśpionych" serwerów. Generalnie, większość z rozpatrywanych wariantów należy do klasy problemów *NP-trudnych*.

W pracy rozpatrujemy system, w którym generowane zbiory danych muszą zostać zreplikowane, w postaci zapasowej kopii, w innej lokalizacji dostępnej za pośrednictwem łącza (kanału) o ograniczonej przepustowości. Kopię należy wykonać, w z góry ustalonym czasie od momentu jej wygenerowania. Niedotrzymanie tego terminu wiąże się z karą, której wielkość zależy od czasu spóźnienia oraz określonego współczynnika kary. Specyfika pracy systemu nie pozwala na wykonywanie kopii *on-line*, gdyż wiązałoby się to z wykorzystaniem zasobów zarezerwowanych do wykonywania bieżących zadań zlecanych przez użytkowników systemu (np. kompresja i szyfrowanie pewnych fragmentów, zajętości kanału, itp.). Dlatego też zadania wykonania kopii zapasowej są kolejkowane i uruchamiane w chwili mniejszej aktywności użytkowników. Należy zatem przydzielić zadania wykonania kopii do konkretnych lokalizacji (kanałów) oraz dla każdej z nich ustalić kolejność wysyłania tak, aby zminimalizować sumę kar za przekroczenie ustalonych terminów.

2. Opis problemu

Dla uproszczenia zapisu, przesyłany zbiór danych będziemy nazywali informacją. Rozpatrywany w pracy problem wykonania kopii można przedstawić następująco.

Dany jest zbiór n informacji $\mathcal{I} = \{1, 2, \ldots, n\}$, które należy przesłać kanałami z m elementowego zbioru $\mathcal{K} = \{1, 2, \ldots, m\}$. Dla informacji $i \in \mathcal{I}$, niech p_i będzie *normatywnym czasem przesyłu*, d_i *żądanym terminem dostarczenia*, a w_i *wagą funkcji kosztów*. Przez v_j ($v_j \geqslant 1$) oznaczamy *współczynnik przepustowości kanału* $j \in \mathcal{K}$. Dla kanału o najmniejszej przepustowości współczynnik ten jest równy 1. Wobec tego, rzeczywisty czas przesyłania informacji $i \in \mathcal{I}$ przez kanał $j \in \mathcal{K}$ wynosi więc $p_i \cdot v_j$. Rozpatrywany problem polega na przydzieleniu informacji do kanałów oraz wyznaczeniu kolejności ich wysyłania, w każdym z kanałów, aby zminimalizować sumę kosztów spóźnień (w skrócie problem ten będziemy oznaczali przez MKS). Dla ustalenia uwagi zakładamy, że przesyłanie informacji rozpoczyna sie w chwili 0.

Dla ustalonych przydziałów informacji do kanałów oraz ustalonych kolejnościach ich wysyłania, przez C_i oznaczamy termin zakończenia przesyłania informacji $i \in \mathcal{I}$. Wówczas $T_i = \max\{0, C_i - d_i\}$ jest *spóźnieniem*, a $w_i \cdot T_i$ *kosztem spóźnienia*. Należy wyznaczyć taki przydział oraz kolejności wysyłania, aby zminimalizować sumę kosztów spóźnień (*całkowitą karę*), tj. sumę $\sum_{i=1}^{n} w_i \cdot T_i$. Muszą być przy tym spełnione następujące ograniczenia:

(*i*) wysyłanie dowolnej informacji nie może być przerwane,

(*ii*) w dowolnym momencie przez kanał może być wysyłana co najwyżej jedna informacja,

(*iii*) dowolna informacja może być przesłana tylko przez jeden kanał.

Z powyższych ograniczeń wynika, że $S_i = C_i - p_i$ jest *momentem rozpoczęcia wysyłania* i*-tej informacji*.

Dla zbioru informacji $\mathcal{I} = \{1, 2, \ldots, n\}$ oraz kanałów $\mathcal{K} = \{1, 2, \ldots, m\}$, ciąg zbiorów informacji

$$Q = (Q_1, Q_2, \ldots, Q_m),$$

takich, że

$$Q_i \cap Q_j = \emptyset,\ i \neq j,\ i, j \in \mathcal{K} \text{ oraz } \sum_{i=1}^{m} Q_i = \mathcal{I}$$

nazywamy *przydziałem informacji do kanałów*. Przez $\mathcal{Q}$ oznaczamy zbiór wszystkich takich przydziałów.

Dla przydziału $Q = (Q_1, Q_2, \ldots, Q_m)$, $Q \in \mathcal{Q}$, niech

$$\pi = (\pi_1, \pi_2, \ldots, \pi_m)$$

będzie ciągiem permutacji takim, że π_i jest n_i elementową ($n_i = |Q_i|$) permutacją (kolejnością wysyłania) informacji ze zbioru Q_k, $k \in \mathcal{K}$, tj. przez k-ty kanał. Przez $\mathcal{P}(Q)$ oznaczamy zbiór wszystkich takich ciągów permutacji (w skrócie ciągi te będziemy nazywali permutacją).

Zbiór rozwiązań dopuszczalnych problemu rozsyłania informacji w sieci można zdefiniować następująco:

$$\mathcal{QP} = \{(Q, \pi) :\ Q \in \mathcal{Q} \ \wedge\ \pi \in \mathcal{P}(Q)\}. \tag{1}$$

Niech $\Theta = (Q, \pi) \in \mathcal{QP}$ będzie pewnym rozwiązaniem, gdzie $Q = (Q_1, Q_2, \ldots, Q_m)$ oraz $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$. Jeżeli informacja $i \in \mathcal{I}$ jest wysyłana przez kanał $k \in \mathcal{K}$ (tj. $i \in Q_k$) jako l-ta w kolejności ($\pi_k(l) = i$), to moment zakończenia jej przesyłu $C_i = \sum_{j=1}^{l} p_{\pi_k(j)} \cdot v_{\pi_k(j)}$, spóźnienie $T_i = \max\{0, C_i - d_i\}$, a koszt spóźnienia $f_i(\Theta) = w_i \cdot T_i$. Wówczas

$$F_k(\Theta) = \sum_{j \in Q_k} f_j(\Theta), \tag{2}$$

jest kosztem przesłania informacji kanałem k-tym, a

$$F(\Theta) = \sum_{j=1}^{n} f_j(\Theta), \tag{3}$$

kosztem wysłania wszystkich informacji (sumą kar za wszystkie spóźnienia).

Wobec tego, rozpatrywany w pracy problem rozsyłania informacji polega na wyznaczeniu rozwiązania $\Theta^* \in \mathcal{QP}$ takiego, że

$$F(\Theta^*) = \min\{F(\Theta) :\ \Theta \in \mathcal{QP}\}. \tag{4}$$

3. Metoda rozwiązania

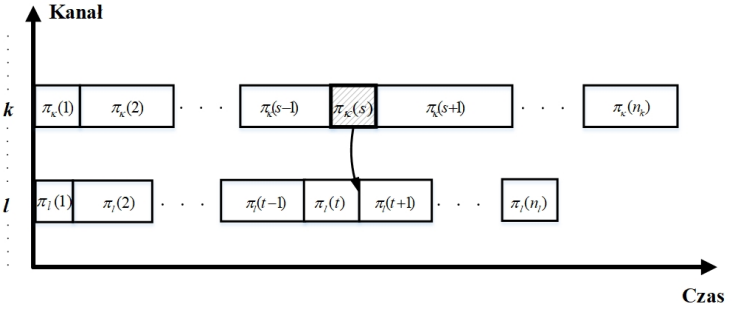
Dla ustalonego przydziału informacji do kanałów, wyznaczenie optymalnej (tj. minimalizującej sumę kosztów spóźnień) kolejności ich wysyłania przez pojedynczy kanał jest już problemem NP-trudnym i sprowadza się do rozwiązania pewnego jednomaszynowego problemu szeregowania zadań oznaczanego w literaturze przez $1||\sum w_iT_i$. Wobec tego, do rozwiązania problemu MKS będziemy stosowali szybki algorytm konstrukcyjny, którego rozwiązanie będzie punktem startowym dla algorytmu przeszukiwania z tabu (ang. *tabu search*).

Niech $\Theta = (Q, \pi)$ będzie pewnym rozwiązaniem problemu MKS. Rozpatrujemy dwa kanały k i l $(k \neq l,\ k, l \in \mathcal{K})$. Z rozwiązania Θ generujemy nowe rozwiązanie $\Theta' = (Q', \pi')$ przez przeniesienie jednej informacji z kanału k do l. Niech s będzie pewną pozycją w permutacji π_k, a t pozycją w permutacji π_l. Ruch typu *transfer* (w skrócie τ - *ruch*) $\tau_l^k(s,t)(\Theta)$ przenosi informację znajdującą się na pozycji s w kanale k na pozycję t w kanale l generując w ten sposób nowe rozwiązanie Θ' (w skrócie będziemy pisali, że $\Theta' = \tau_l^k(s,t)(\Theta)$). Dokładniej, niech $\Theta = (Q, \pi)$, gdzie $Q = (Q_1, Q_2, \ldots, Q_m)$, $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, a wygenerowane rozwiązanie $\Theta' = (Q', \pi')$, przy czym $Q' = (Q'_1, Q'_2, \ldots, Q'_m)$, $\pi' = (\pi'_1, \pi'_2, \ldots, \pi'_m)$. Wówczas

$$Q'_i = Q_i \ \wedge \ \pi'_i = \pi_i, \quad i \neq k, l \ (k, l \in \mathcal{K}), \tag{5}$$

$$\begin{gathered} Q'_k = Q_k \ \setminus \{\pi_k(s)\},\ Q'_l = Q_l \cup \{\pi_k(s)\}, \\ \pi'_k = (\pi_k(1), \pi_k(2), \ldots, \pi_k(s-1), \pi_k(s+1), \ldots, \pi_k(n_k)), \\ \pi'_l = (\pi_l(1), \pi_l(2), \ldots, \pi_l(t-1), \pi_k(s), \pi_l(t), \pi_l(t+1) \ldots, \pi_l(n_l)), \\ n'_k = n_k - 1,\ n'_l = n_l + 1. \end{gathered} \tag{6}$$

Łatwo sprawdzić, że tak wygenerowane rozwiązanie $\Theta' = (Q', \pi')$ jest dopuszczalnym dla problemu MKS. Symbolicznie generowanie rozwiązania Θ' jest przedstawione na rysunku 1.



Rys. 1. Transfer informacji $\pi_k(s)$ z kanału k do l.

Zbiór rozwiązań

$$N_l^k(\Theta) = \{\tau_l^k(s,t)(\Theta) :\ s = 1, 2, \ldots, n_k,\ t = 1, 2, \ldots, n_l\}, \tag{7}$$

będziemy nazywali *otoczeniem* kanałów k i l. Liczba elementów tego otoczenia wynosi $n_k \cdot (n_l + 1)$. W dalszej części tego rozdziału zakładamy, że rozwiązanie $\Theta = (Q, \pi)$, a rozwiązanie $\Theta' = \tau_l^k(s,t)(\Theta) = (Q', \pi')$, gdzie $Q = (Q_1, Q_2, \ldots, Q_m)$, $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ oraz $Q' = (Q'_1, Q'_2, \ldots, Q'_m)$, $\pi' = (\pi'_1, \pi'_2, \ldots, \pi'_m)$.

Niech

$$\tilde{F}_k(\Theta') = \sum_{i=1}^{n'_k} f_{\pi'_k(i)}(\Theta'). \tag{8}$$

Jest to koszt wysłania informacji przez k-ty kanał w kolejności π'_k (tj. po usunięci z permutacji π_k informacji $\pi_k(s)$).
Dla rozwiązania $\Theta \in \mathcal{QP}$, niech

$$\delta^-_\Theta = \sum_{i=s+1}^{n_k-1} w_{\pi_k(i)} \cdot \max\{0, C_{\pi_k(i+1)} - v_{\pi_k(s)} \cdot p_{\pi_k(s)} - d_{\pi_k(i+1)}\} -$$

$$-(f_{\pi_k(s)}(\Theta) + \sum_{i=s+1}^{n_k} f_{\pi_k(i)}(\Theta)). \tag{9}$$

Lemat 1. Jeżeli rozwiązanie $\Theta' = \tau^k_l(s,t)(\Theta)$, to

$$\tilde{F}_k(\Theta') = F_k(\Theta) + \delta^-_\Theta$$

jest górnym ograniczeniem optymalnej wartości kosztu wysyłania informacji przez k-ty kanał.
Dowód. Ponieważ

$$F_k(\Theta) = \sum_{i=1}^{s-1} f_{\pi_k(i)}(\Theta) + f_{\pi_k(s)} + \sum_{i=s+1}^{n_k} f_{\pi_k(i)}(\Theta),$$

więc

$$F_k(\Theta) + \delta^-_\Theta = \sum_{i=1}^{s-1} f_{\pi_k(i)}(\Theta) + f_{\pi_k(s)} + \sum_{i=s+1}^{n_k} f_{\pi_k(i)}(\Theta) +$$

$$+ \sum_{i=s+1}^{n_k-1} w_{\pi_k(i)} \cdot \max\{0, C_{\pi_k(i+1)} - v_{\pi_k(s)} \cdot p_{\pi_k(s)} - d_{\pi_k(i+1)}\} - (f_{\pi_k(s)}(\Theta) + \sum_{i=s+1}^{n_k} f_{\pi_k(i)}(\Theta))$$

$$= \sum_{i=1}^{s-1} f_{\pi_k(i)}(\Theta) + \sum_{i=s+1}^{n_k-1} w_{\pi_k(i)} \cdot \max\{0, C_{\pi_k(i+1)} - v_{\pi_k(s)} \cdot p_{\pi_k(s)} - d_{\pi_k(i+1)}\} = F_k(\Theta'),$$

co kończy dowód lematu. Podobny lemat można udowodnić dla l-tego kanału.

Niech

$$\delta^+_\Theta = w_{\pi_{\pi_l(s)}} \cdot \max\{0, C_{\pi_l(t-1)} + v_{\pi_k(s)} \cdot p_{\pi_k(s)} - d_{\pi_k(s}\} + \tag{10}$$

$$+ \sum_{i=t}^{n_l} w_{\pi_k(i)} \cdot \max\{0, C_{\pi_l(i)} + v_{\pi_k(s)} \cdot p_{\pi_k(s)} - d_{\pi_l(i)}\} - \sum_{i=t}^{n_l} f_{\pi_l(i)}(\Theta).$$

Lemat 2. Jeżeli rozwiązanie $\Theta' = \tau^k_l(s,t)(\Theta)$, to

$$\tilde{F}_l(\Theta') = F_l(\Theta) + \delta^+_\Theta$$

jest górnym ograniczeniem optymalnej wartości kosztu wysyłania informacji przez l-ty kanał.
Dowód. Jest podobny do dowodu Lematu 1.

Niech

$$\Delta_l^k(s,t)(\Theta) = \delta_\Theta^- + \delta_\Theta^+. \tag{11}$$

Wartość tego wyrażenia można wyznaczyć w czasie $O(n)$. Będzie ono stosowane do szacowania efektywności ruchu typu transfer (tj. wartość górnego ograniczenia funkcji celu dla generowanego przez ten ruch rozwiązania). Pozwoli to wyznaczyć z otoczenia najbardziej obiecujące rozwiązanie.

Twierdzenie 1. Jeżeli Θ jest pewnym rozwiązaniem problemu MKS oraz Θ' zostało wygenerowane z Θ przez wykonanie ruchu transfer $\tau_l^k(s,t)$, to górne ograniczenie kosztu wysyłania wszystkich informacji

$$\tilde{F}(\Theta') = F(\Theta) + \Delta_l^k(s,t)(\Theta). \tag{12}$$

Dowód. Dowód wynika bezpośrednio z Lematu 1 i 2.

Przy wyborze elementu z otoczenia $\mathcal{N}_l^k(\Theta)$ będziemy korzystali z wyrażenia $\Delta_l^k(s,t)(\Theta)$, tj. wybieramy z otoczenia element, dla którego jego wartość jest minimalna. Jest to więc element o najmniejszej wartości górnego ograniczenia funkcji kryterialnej.

3.1. Algorytmy rozwiązywania problemu

W tym rozdziale przedstawiamy dwa algorytmy rozwiązywania rozpatrywanego w pracy problemu replikacji zbiorów. Algorytm konstrukcyjny bazujący na metodzie zachłannej z wykorzystaniem idei algorytmu NEH (Nawaz, Enscore, Ham [8]) oraz algorytm przeszukiwania z tabu.

ALGORYTM GREENNEH (*algorytm zachłanny z elementami NEH*)

Wyjście: $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ – kolejności wysyłania informacji;

1. **begin**
2. $\pi_l \leftarrow (), n_l := 0, l = 1, 2, \ldots, m;$
3. Ponumerować informacje tak, aby $d_1 \leqslant d_2 \leqslant, \ldots, \leqslant d_n$:
4. **for** $i := 1$ **to** n **do**
5. **Step 1:** Wyznaczyć kanał l o najmniejszym koszcie
6. wysyłania informacji;
7. Wstawić na próbę informację i na każdą z pozycji
8. $1, 2, \ldots, n_l + 1$ w permutacji π_l;
9. Wybrać pozycję t, na której koszt wysyłania wszystkich informacji
10. (z π_l wraz z informacją i) jest minimalny;
11. **Step 2:** Wstawić i na pozycję t do π_l oraz $n_l := n_l + 1$;
12. **end.**

W linii 5 i 6 zastosowano zmodyfikowaną strategie BF stosowaną w algorytmach pakowania. Z kolei w opisie zamieszczonym w liniach 7-10 wykorzystano idee stosowaną w bardzo dobrym algorytmie konstrukcyjnym NEH do wyznaczania rozwiązania przybliżonego dla problemu przepływowego z kryterium $C_{\max}$. Średnia złożoność obliczeniowa procedury sortowania (Quicksort, linia 3) wynosi $O(nlgn)$. Wybór odpowiedniego kanału (linie 5-6) jest rzędu $O(m)$, a wyznaczenie odpowiedniej pozycji (linie

7-10), na której będzie umieszczona wstawiana informacja ma złożoność $O(n^2)$. Tak więc algorytm **GRENNEH** ma złożoność $O(n^2)$. Na podstawie wyznaczonej przez algorytm permutacji π łatwo można wyznaczyć odpowiadający jej przydział informacji do kanałów.

Punktem startowym drugiego algorytmu, przeszukiwania z tabu, jest rozwiązanie Θ wyznaczone przez algorytm konstrukcyjny **GRENNEH**.

ALGORYTM ATS

1. $\Theta^* := \Theta$;
2. **repeat**
3. wyznacz numery kanałów k i l
   (z kanału k pewna informacja zostanie przeniesiona do kanału l);
4. wygeneruj otoczenie $\mathcal{N}_l^k(\Theta)$;
5. wyznacz z otoczenia rozwiązanie Θ' o minimalnej wartości
6. górnego ograniczenia funkcji kosztu;
7. **if** $F(\Theta') < F(\Theta^*)$ **then** $\Theta^* := \Theta'$;
8. $\Theta := \Theta'$
9. **until** *warunek zatrzymania*;
10. Zoptymalizuj, w każdym z kanałów, kolejność wysyłania informacji.

Warunkiem zatrzymania może być np. z góry ustalona liczba iteracji. W linii 3. algorytmu, kanały k i l mogą być wyznaczone losowo lub k może być kanałem o największej karze, a l o najmniejszej. W linii 10. można zastosować dowolny algorytm rozwiązania *NP*-trudnego problemu $1||\sum w_i T_i$. W naszej implementacji będziemy stosowali metaheurystykę przeszukiwania z tabu zamieszczoną w pracy [3].

4. Eksperymenty obliczeniowe

Przedstawione w pracy algorytmy zostały zaimplementowane w języku C# oraz uruchomione na komputerze osobistym z procesorem Intel i7. Liczba iteracji algorytmu **ATS** (*maxit*=100). Eksperymenty obliczeniowe wykonano na przykładach wygenerowanych losowo zgodnie z procedurą opisaną w *OR-Library* [1]. Czasy przesyłu p_i są realizacją zmiennej losowej o rozkładzie jednostajnym na przedziale [1,100], a wagi w_i oraz współczynniki przepustowości v_i na przedziale [1,10]. Wartości żądanych terminów zakończenia d_i zależą od dwóch parametrów RDD (ang. relative of due date) oraz TF (ang. average tardiness factor), które przyjmują wartości ze zbioru $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Terminy te, podobnie jak pozostałe wartości, są losowane z przedziałów [P(1-TF-RDD/2),P(1-TF+RDD/2)], gdzie $P = \sum p_i$. Dla tych samych wartości parametrów RDD i TF generowano po dwa przykłady. Dla małych wartości $\{0.2, 0.4\}$ przykłady są uważane za "łatwe", a dla dużych $\{0.8, 1.0\}$ - "trudne". Porównano wartości wyznaczanych przez oba algorytmy rozwiązań. Procentowy błąd względny (dla czasów obliczeń lub wartości rozwiązań) wyznaczano z następującego wzoru:

$$PRD = \frac{(\mathcal{F}_{GRENNEH} - \mathcal{F}_{ATS})}{\mathcal{F}_{ATS}} \cdot 100\%. \tag{13}$$

Obliczenia wykonano dla liczby informacji $n = 10, 500, 1000$ oraz kanałów $m = 2, 5$. W Tabeli 1 przedstawiamy wyniki dla przykładów rozsyłania 1000 informacji po-

przez dwa kanały. Wyniki uzyskane dla pozostałych przykładów były podobne. Średnia

Tabela 1
Porównanie wyników algorytmów **GRENNEH** oraz **ATS** ($n = 1000,\ m = 2$).

Przykład	$\mathcal{F}_{GRENNEH}$	$\mathcal{F}_{ATS}$	PRD
1	1532172	1045732	31,75
2	1701543	1209850	28,90
3	1656852	1495432	9,74
4	1346486	908754	32,51
5	1486194	1109388	25,35
6	1612400	1275481	20,90
7	1259705	897653	28,74
8	1624126	1109650	31,68
9	1722109	1108598	35,63
10	1358982	797651	41,31
Średnio			28,65

względna poprawa wartości rozwiązania wynosi 28% (dla przykładów "trudnych" jest to prawie 40%). Zastosowanie w algorytmie **ATS** szacowania wartości funkcji kryterialnej (Twierdzenie 1) skraca kilkukrotnie czas obliczań, bez zmiany średniej wartości wyznaczanych rozwiązań.

5. Podsumowanie

W pracy jest rozpatrywany problem replikacji zbiorów danych w chmurze obliczeniowej. Należy on do klasy problemów NP-trudnych. Przedstawiono metodę szacowania funkcji kryterialnej umożliwiającą przyspieszenie procedury przeszukiwania otoczenia w algorytmie TS. Przeprowadzono eksperymenty obliczeniowe na losowo generowanych przykładach. Wykazały one znaczną poprawę wartości rozwiązań przez algorytm TS. Dalsze badania będą miały na celu wskazanie obiecujących obszarów przestrzeni rozwiązań, które będą przeszukiwane.

Praca powstała w wyniku realizacji projektu badawczego o nr DEC 2017/25/B/ST7 /02181 finansowanego ze środków Narodowego Centrum Nauki.

LITERATURA

1. Beasley E.: Weighted tardiness, OR-Library, http://people.brunel.ac.uk/ mastjjb/jeb/info.html
2. Balasaraswathi V.R., Manikandan S.: Enhanced security for multi-cloud storage using cryptographic data splitting with dynamic approach. ICACCCT Conference, 2004, p. 1190–1194.
3. Bożejko W., Grabowski J., Wodecki M.: Block approach-tabu search algorithm for single machine total weighted tardiness problem, Computers & Industrial Engineering, 50(1/2), 2006, p. 1–14.

4. Djebbar E.I., Belalem G., Benadda M.: Task scheduling strategy based on data replication in scientific Cloud workflows, Multiagent and Grid Systems 12(1), 2016, p. 55–67.
5. Hadji M.: Scalable and Cost-Efficient Algorithms for Reliable and Distributed Cloud Storage. In: International Conference on Cloud Computing and Services Science. Springer International Publishing, 2015, p. 15–37.
6. Liu, G., Shen, H., Yu, L.:Towards deadline guaranteed cloud storage services, IEEE International Conference on Cloud Computing, CLOUD 7820274, 2017, p. 212–219.
7. Mansouri Y., Toosi A.N., Buyya R.: Brokering algorithms for optimizing the availability and cost of cloud storage services. International Conference on Cloud Computing Technology and Science, vol. 01, 2013, p. 581–589.
8. Nawaz M., Enscore E.E., Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA, 11/1, 1983, p. 91–95.
9. Thanasis P.G., Bonvin N., Aberer K.: Scalia: an adaptive scheme for efficient multicloud storage. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Los Alamitos, USA, 2012, p. 1–20.
10. Yanzhen Q., Naixue X.: RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. In: Parallel Processing (ICPP) Conference, 2012, p. 520–529.