

ALGORYTM EWOLUCYJNY DLA PROBLEMU SZEREGOWANIA ZADAŃ W SYSTEMIE PRZEPŁYWOWYM

Adam STAWOWY, Marek ŚWIĘCHOWICZ

Streszczenie: W pracy zaprezentowano algorytm strategii ewolucyjnej do problemu szeregowania zadań w permutacyjnym systemie przepływowym przy kryterium minimalizacji długości uszeregowania oraz przedstawiono wyniki badań porównawczych z techniką Tabu Search. W opracowanej przez autorów wersji tego algorytmu wykorzystano nowy operator mutacji dostosowany do problemu szeregowania. Wyniki badań wykazały, że algorytm ten, szczególnie dzięki nowej mutacji oraz starannemu doborowi wszystkich parametrów, jest dokładniejszy niż algorytmy genetyczne opisywane w literaturze i może konkurować, z dużo bardziej skomplikowanymi, algorytmami Tabu Search.

Słowa kluczowe: szeregowanie zadań, problem przepływowy, algorytmy ewolucyjne

1. Wprowadzenie

Zagadnienie szeregowania zadań w permutacyjnym systemie przepływowym jest problemem silnie NP-trudnym, co oznacza, że nie jest możliwe sformułowanie efektywnego algorytmu, dla którego funkcja złożoności obliczeniowej jest ograniczona od góry przez wielomian liczby zadań n [1]. Dlatego też, już od połowy lat 60, obserwuje się burzliwy rozwój wielomianowych algorytmów heurystycznych dających rozwiązania przybliżone.

Jako kryterium optymalizacji w procesie przepływowym stosowana jest powszechnie *długość uszeregowania* (DU) czyli czas zakończenia wykonywania zadań w systemie (ang. makespan):

$$C_{\max} = \max_i (C_{iM}) \quad (1)$$

gdzie:

- n - liczba zadań,
- M - liczba stopni przetwarzania (zespołów technologicznych, maszyn, urządzeń),
- M_j - urządzenie nr j , $j=1 \dots M$,
- Z_i - zadanie nr i , $i=1 \dots n$,
- t_{ij} - czas wykonywania operacji zadania Z_i na urządzeniu M_j ,
- C_{ij} - czas zakończenia wykonywania zadania Z_i na urządzeniu M_j .

2. Strategia ewolucyjna do szeregowania zadań w systemie przepływowym

W ostatnich latach pojawiło się kilka publikacji dotyczących zastosowania algorytmów ewolucyjnych dla problemu przepływowego - wszystkie wykorzystują różne wersje algorytmu genetycznego: są to przede wszystkim prace Mulkensa [2], Reeves'a [3] oraz Chena i zespołu [4]. W publikacjach autorzy ci porównali swoje heurystyki z techniką symulowanego wyżarzania oraz algorytmem Ho i Changa [5]; badania obliczeniowe potwierdziły efektywność działania techniki GA.

Wersja strategii ewolucyjnej zastosowana przez autorów dla szeregowania zadań w systemie przepływowym jest wzorowana na algorytmie Nissena dla problemu alokacji zasobów

[6]. Jest to $(1,\lambda)$ -ES, w której λ potomków jest generowanych z jednego rodzica za pomocą prostej mutacji. Krzyżowanie nie jest stosowane. Najlepszy z potomków zastępuje rodzica w nowej populacji. Ten sposób selekcji (rodzic nie konkuruje z potomkami) powoduje często pogorszenie rozwiązania wejściowego, ale - wg Nissena - poprawia działanie algorytmu (potwierdziły to również obserwacje autorów).

Ochronę przed przedwczesną zbieżnością algorytmu do optimum lokalnego stanowi mechanizm tzw. destabilizacji. Jeśli wartość funkcji dopasowania najlepszego potomka jest mniejsza niż wartość funkcji rodzica, zmienna (licznik) zliczająca takie zdarzenia jest zwiększana o jeden, w przeciwnym razie - jest zerowana. Jeżeli licznik przekroczy zadaną wartość, następuje faza destabilizacji, podczas której jest zerowany licznik i następuje generowanie potomków poprzez intensywną mutację rodzica. W naszym algorytmie przyjęto, że jest nią mutacja zamiany wykonywana 2000 razy dla ostatniego rozwiązania rodzicielskiego. W ten sposób powstaje potomek bardziej różniący się od rodzica niż to ma miejsce podczas normalnego działania algorytmu, a przeszukiwanie zostaje skierowane w nowe obszary przestrzeni dopuszczalnych rozwiązań.

Eksperymentalnie ustalono następujące elementy algorytmu:

- rozmiar populacji,
- parametry mechanizmu destabilizacji,
- warunek zakończenia obliczeń.

Przyjęto, że funkcja dopasowania jest określona następującym wzorem:

$$FD = \frac{DG}{1 + DU - DG} \quad (2)$$

gdzie:

DG - dolna granica obliczana wg zasady podanej w pracy Taillarda [7],

DU - długość uszeregowania (makespan).

Taka konstrukcja funkcji pozwala na jej maksymalizowanie, co jest zalecane dla poprawnego działania algorytmów ewolucyjnych. Ponadto funkcja taka silniej różnicuje rozwiązania o zbliżonej długości uszeregowania niż czyniłaby to funkcja jedynie długości uszeregowania oraz niesie dodatkową informację o potencjalnie optymalnym rozwiązaniu. Przyjęta funkcja ma również tę właściwość, że jeśli $DU=DG$ to $FD=DG$.

Operator mutacji wzięty z typowego algorytmu ewolucyjnego nie jest odpowiedni dla zagadnień permutacyjnych [8]. Autorzy opracowali nowy operator bazujący na założeniach, że potomek nie powinien różnić się bardzo od rodzica oraz że intensywność mutacji powinna maleć w kolejnych pokoleniach. W rezultacie powstał operator NDSM (ang. Normally Distributed Shift Mutation) działający w ten sposób, że zadanie z wylosowanej pozycji przesuwane jest o liczbę pozycji wylosowaną z rozkładu normalnego $N(0,\sigma)$.

Formalny zapis działania operatora NDSM wygląda następująco:

$$\text{przesunięcie} = \text{round}(N(0,\delta))$$

$$\delta = \delta \bullet cr^{gen}$$

(3)

gdzie:

σ_0 - początkowe odchylenie standardowe równe $0.15 \cdot n$,

- cr - czynnik redukujący wartość odchylenia w kolejnych generacjach
 $cr = \exp((\ln(\sigma_k) - \ln(\sigma_0)) / maxgen,$
 σ_k - końcowa wartość odchylenia standardowego równa $0.03 * n$,
 gen - numer aktualnej generacji,
 $maxgen$ - maksymalna liczba generacji.

Intensywność mutacji (mierzona średnią wielkością przesunięcia) jest duża na początku działania algorytmu i zmniejsza się z czasem w ślad za zmianą wartości odchylenia standardowego, które maleje wykładniczo od σ_0 do σ_k . Jest to intuicyjnie zrozumiałe, gdyż na początku procesu przeszukiwania szansa na poprawę rodzicielskiego rozwiązania poprzez silniejsze zaburzenie jest dużo większa niż w końcowej fazie procesu, gdzie rodzic jest z reguły rozwiązaniem bliskim optymalnemu.

Ostateczną wersję strategii ewolucyjnej dla problemu przepływowego charakteryzują zebrane w tabeli 1 elementy i parametry.

Tab. 1. Elementy strategii ewolucyjnej dla problemu przepływowego

Funkcja dopasowania	
Postać	$FD(R_i) = \frac{DG}{1 + DU(R_i) - DG}$
Techniki zaawansowane	nie stosowano
Schemat i populacja	
Reprezentacja	lista (ciąg) n zadań
Rozmiar populacji	1 rodzic, 35 potomków
Inicjalizacja	1 rozwiązanie wyznaczone algorytmem NEH [9]
Selekcja następnego pokolenia	najlepszy potomek zastępuje rodzica
Warunek zatrzymania	po wygenerowaniu 35 000 pokoleń
Mechanizm destabilizacji	po 900 kolejnych generacjach, podczas których nie nastąpiła poprawa najlepszego znalezionej przez algorytm rozwiązania
Reprodukcja	nie stosowano
Mutacja	
Operator mutacji	NDSM
P_m	1,0

3. Badania eksperymentalne

Algorytm ewolucyjny testowano przy użyciu przykładowych problemów Taillarda [7]. Testy te składają się ze 120 szczególnie trudnych przypadków o 12 różnych rozmiarach, wybranych spośród wielkiej liczby losowo wygenerowanych problemów. Dla każdego $n * M = \{20 * 5, 20 * 10, 20 * 20, 50 * 5, 50 * 10, 50 * 20, 100 * 5, 100 * 10, 100 * 20, 200 * 10, 200 * 20, 500 * 20\}$ Taillard wybrał 10 przykładów.

Algorytm strategii ewolucyjnej (ES) porównano z algorytmami Tabu Search Taillarda (T) [10] oraz Nowickiego i Smutnickiego (TSAB) [11]. Algorytm ewolucyjny działał zgodnie ze schematem przedstawionym w tabeli 1. W każdym przypadku wykonywano pięć przebiegów obliczeniowych, spośród których wybrano najlepszy rezultat. Wyniki testów zawiera tabela 2; pogrubioną czcionką oznaczono najlepsze rezultaty spośród otrzymanych podczas badań lub znalezionych w literaturze. Nie przedstawiono wyników testów dla problemów o

najmniejszych rozmiarach {20*5, 20*10, 20*20, 50*5}, gdyż wszystkie algorytmy osiągnęły identyczne rezultaty; z tego też względu nie uwzględniono tych wyników w badaniach statystycznych.

Następnie zbadano, czy różnice pomiędzy wynikami otrzymywanymi przez poszczególne techniki są istotne. W tym celu dla każdego rozmiaru problemu obliczono dla poszczególnych algorytmów względną odległość od najlepszego rozwiązania i tak otrzymane średnie poddano testowi Cochran-Coxa [12].

Tab. 2. Zestawienie długości uszeregowień porównywanych algorytmów dla przykładowych problemów Taillarda

Rozmiar	50*10			50*20		
Algorytm	T	TSAB	ES	T	TSAB	ES
1	3037	3025	3025	3886	3875	3874
2	2911	2892	2885	3733	3715	3718
3	2871	2864	2862	3673	3668	3669
4	3067	3064	3063	3755	3752	3763
5	3011	2986	2984	3648	3635	3649
6	3021	3006	3006	3719	3698	3716
7	3124	3107	3111	3730	3716	3731
8	3048	3039	3039	3737	3709	3729
9	2910	2902	2902	3772	3765	3782
10	3100	3091	3091	3791	3777	3786
Rozmiar	100*5			100*10		
Algorytm	T	TSAB	ES	T	TSAB	ES
1	5493	5493	5493	5776	5770	5771
2	5274	5268	5268	5362	5349	5361
3	5175	5175	5175	5679	5677	5679
4	5018	5014	5018	5820	5791	5791
5	5250	5250	5252	5491	5468	5475
6	5135	5135	5135	5308	5303	5308
7	5247	5246	5251	5600	5599	5643
8	5094	5094	5094	5640	5623	5637
9	5448	5448	5448	5891	5875	5884
10	5328	5328	5322	5865	5845	5881
Rozmiar	100*20			200*10		
Algorytm	T	TSAB	ES	T	TSAB	ES
1	6330	6286	6288	10872	10868	10872
2	6320	6241	6255	10500	10494	10506
3	6364	6329	6339	10956	10922	10948
4	6331	6306	6366	10893	10889	10893
5	6405	6377	6424	10537	10524	10537
6	6487	6437	6443	10347	10331	10347
7	6379	6346	6338	10882	10857	10882
8	6514	6481	6509	10754	10731	10753
9	6386	6358	6358	10465	10438	10443
10	6534	6465	6525	10727	10676	10685

Rozmiar	200*20			500*20		
Algorytm	T	TSAB	ES	T	TSAB	ES
1	11393	11294	11354	26316	26189	26259
2	11445	11420	11391	26807	26629	26809
3	11522	11446	11496	26626	26458	26596
4	11461	11347	11435	26642	26549	26646
5	11427	11311	11384	26509	26404	26473
6	11368	11282	11325	26654	26581	26606
7	11536	11456	11476	26575	26461	26577
8	11544	11415	11516	26794	26615	26762
9	11424	11343	11382	26241	26083	26226
10	11528	11422	11465	26662	26527	26599

Analizując otrzymane wyniki można stwierdzić, iż nie ma statystycznych różnic między algorytmami dla problemów o rozmiarach 100*5 i 100*10. W pozostałych przypadkach:

- algorytm TSAB jest lepszy niż technika TS w wersji Taillarda,
- TSAB dominuje nad strategią ewolucyjną dla problemów o większych rozmiarach ($n \geq 200$),
- strategia ewolucyjna jest bardziej efektywna niż algorytm Taillarda dla problemów 50*10 i 200*20; dla pozostałych problemów nie ma statystycznych różnic między tymi technikami.

Warto zauważyć, że w niektórych przypadkach bardzo dobre rezultaty dała prosta strategia ewolucyjna, która ogółem znalazła osiem rozwiązań lepszych niż spotkane do tej pory w literaturze fachowej.

4. Wnioski końcowe

Badania eksperymentalne wykazały, że:

- algorytm ewolucyjny opracowany przez autorów, szczególnie dzięki nowej mutacji oraz starannemu doborowi wszystkich parametrów, jest efektywniejszy niż algorytmy genetyczne opisywane w literaturze i może konkurować, z dużo bardziej skomplikowanymi, algorytmami typu Tabu Search,
- istnieje możliwość wykorzystania opisanego algorytmu w systemach, gdzie czynnik czasu ma decydujące znaczenie; algorytm stosunkowo szybko dochodzi do dobrych rozwiązań i w razie konieczności przerywania obliczeń najlepsze znalezione dotychczas rozwiązanie można uznać za rozwiązanie suboptymalne.

Przeprowadzone badania wykazały dużą przydatność algorytmów ewolucyjnych dla problemu szeregowania zadań w permutacyjnym systemie przepływowym. Wielką ich zaletą jest prostota, elastyczność oraz brak potrzeby wnikania w strukturę rozwiązywanego problemu.

Literatura

1. Błażewicz J., Cellary W., Słowiński R., Węglarz J.: Badania operacyjne dla informatyków. WNT, Warszawa, 1983.
2. Mulken H.: Revisiting the Johnson algorithm for flow-shop scheduling with genetic algorithms. [w:] Knowledge-Based Reactive Scheduling, North-Holland, Amsterdam, 1994.
3. Reeves C.R., A genetic algorithm for flowshop sequencing. Computers and Operations Research, vol. 22, nr 1, 1994, pp. 5-13.
4. Chen Ch.-L., Vempati V.S., Aljaber N.: An application of genetic algorithms for flow shop problems. European Journal of Operational Research, vol. 80, 1995, pp. 389-396.
5. Ho J.C., Chang Y.L.: A new heuristic for the n-job, M-machine flow-shop problem. European Journal of Operational Research, vol. 52, 1990, pp. 194-206.
6. Nissen V.: Evolutionary facility layout - a comparison. Artykuł zgłoszony do Studies in Locational Analysis, 1997.
7. Taillard E.: Benchmarks for basic scheduling problems. European Journal of Operational Research, vol. 64, 1993, pp. 278-285.
8. Michalewicz Z.: Genetic algorithm + data structures = evolution programs. Springer-Verlag, Berlin, 1992.
9. Nawaz M., Enscore Jr. E.E., Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA International Journal of Management Science, vol. 11, 1983, pp. 91-95.
10. Taillard E.: Some efficient heuristic methods for flow shop sequencing. European Journal of Operational Research, vol. 47, 1990, pp. 65-74.
11. Nowicki E., Smutnicki Cz.: A fast tabu search algorithm for the flow shop problem. Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1994.
12. Domański C.: Testy statystyczne. PWE, Warszawa, 1990.