Problem przydziału dla algorytmów genetycznych

Joanna Szołomicka 244937

Planowanie projektów to ważny proces w każdej firmie. Niewłaściwe zarządzanie zasobami może doprowadzić do przekroczenia czasu przeznaczonego na pracę lub dostępnego budżetu. Problem przydziału polega na rozplanowaniu zadań w projekcie w sposób najbardziej optymalny. Można go rozwiązać za pomocą heurystyk, a w szczególności algorytmów genetycznych.

1 Opis problemu

Problem przydziału ma wiele modyfikacji m.in może dotyczyć jednego lub wielu projektów. Niniejsza praca dotyczy problemu przydziału dla pojedynczego projektu - MRCPSP (multi-mode resource-constrained project sheduling problem) oraz JSP (job scheduling problem).

1.1 JSP [2]

Dane jest $\mathbf{J} = \{1, \ldots, n\}$ prac, które muszą być wykonane przez $\mathbf{m}$ wykonawców. Każda praca składa się z kilku zadań. Zadania należące do pracy j mają określoną kolejność wykonywania daną jako zbiór pierwszeństwa. Każde zadanie jest wykonywane przez jednego wykonawcę w stałym czasie. Wykonawca może wykonywać jedno zadanie na raz, a rozpoczęte zadanie nie może zostać przerwane. Problem polega na znalezieniu kolejności wykonywania zadań, którego czas wykonania jest najkrótszy.

1.2 MRCPSP [1]

Dany jest zbiór prac $\mathbf{J} = \{0, \ldots, n, n+1\}$ oraz zbiór pierwszeństwa zazwyczaj w formie DAG, gdzie krawędź $(\mathbf{u}, \mathbf{v})$ oznacza, że $\mathbf{u}$ musi zakończyć się przed rozpoczęciem $\mathbf{v}$. Prace $\mathbf{0}$ i $\mathbf{n+1}$ są pomocnicze i określają źródło oraz ujście.
Prace $\{1, \ldots, n\}$ zużywają określoną liczbę zasobów odnawialnych oraz nieodnawialnych (np. budżet). Zbiór zasobów odnawialnych dany jest jako K^p, każdy $k \in K^p$ ma określony czas R_k^p - dostępność zasobu k w okresie (do czasu odnowienia). Zbiór zasobów nieodnawialnych dany jest jako K^v, każdy $k \in K^v$ ma określony czas R_k^v - dostępność zasobu k przez cały projekt.
Każde zadanie może zostać wykonane w jednym z kilku trybów. Wykonanie zadania w poszczególnych trybach zużywa różną liczbę zasobów oraz ma różny czas trwania. Praca j może zostać wykonana w jednym z M_j trybów z $\mathbf{M_j} = \{1, \ldots, M_j\}$. Zadanie rozpoczęte w danym trybie nie może zostać przerwane, ani tryb nie może zostać zmieniony.
Dane są:

- p_{jm} - czas wykonania pracy j w trybie m
- r_{jmk}^p - liczba zużytego zasobu odnawialnego $k \in K^p$ podczas wykonywania pracy j w trybie m
- r_{jmk}^v - liczba zużytego zasobu nieodnawialnego $k \in K^v$ podczas wykonywania pracy j w trybie m

Dla $j = 0$ oraz $j = n + 1$:
$p_{jm} = 0$, $r_{jmk}^p = 0$ dla $k \in K^p$ oraz $r_{jmk}^v = 0$ dla $k \in K^v$

Celem jest znalezienie kolejności wykonywania prac w określonych trybach, którego czas wykonania jest najkrótszy.

2 Algorytmy genetyczne

Algorytmy genetyczne wynalezione przez J. Holland w 1970 są zainspirowane biologią populacji oraz genetyką.

Najpierw generowana jest pierwsza populacja, której rozmiar wynosi POP. Następnie wybierane są

Algorithm 1 Algorytm genetyczny

$P_t \leftarrow$ generate initial population
assessFitness(P_t)
while stopping criteria is not satisfied **do**
  select individuals from P_t to P_{t+1}
  crossover individuals from P_t and put into P_{t+1}
  mutate individuals from P_t and put into P_{t+1}
  assessFitness(P_{t+1})
  select POP individuals from $P_t \cup P_{t+1}$ into P_t
end while

osobniki, dzięki którym powstanie kolejna populacja. Osobniki nowych populacji powstają poprzez połączenie chromosomów dwójki rodziców z poprzedniej populacji za pomocą metody *crossover*. Następnie część osobników z nowej populacji jest mutowana przy pomocy metody *mutate*, dzięki czemu zapewniona jest różnorodność genetyczna. Dana jest metoda $assessFitness$, która określa jak dobre geny ma dany osobnik. Nowa populacja może być stworzona jedynie z dzieci ($P_t = P_{t+1}$) lub może być to POP wyselekcjonowanych osobników z populacji o rozmiarze $2 * POP$ ($P_t \cup P_{t+1}$) Proces jest powtarzany dopóki nie zostanie znalezione wystarczająco dobre rozwiązanie lub czas się nie skończy.

3 Rozwiązanie MRCPSP [1]

3.1 Osobniki

Osobnik jest reprezentowany jako para $I = (\lambda, \mu)$, gdzie $\lambda = (j_1, \ldots, j_J)$ jest listą prac, których kolejność jest zgodna z pierwszeństwem prac. μ jest listą trybów w jakich wykonywane są prace j_i.

$$\mathbf{I} = \begin{pmatrix} j_1 & \ldots & j_J \\ \mu(j_1) & \ldots & \mu(j_J) \end{pmatrix}, \mu(j) \in \mathbf{M}_j$$

Każdy osobnik jednoznacznie reprezentuje rozwiązanie problemu. Aby odtworzyć kolejność prac należy zacząć w źródle w czasie 0. Następnie wykonywana jest pierwsza niewykonana praca j_i z listy λ w najwcześniejszym możliwym czasie (muszą być zachowane pierwszeństwo zadań oraz dostępność zasobów odnawialnych) w trybie $\mu(j_i)$.
Dopuszczalne są osobniki, które są niewykonalne względem zasobów nieodnawialnych - znalezienie rozwiązania wykonalnego względem zasobów nieodnawialnych jest problemem NP-trudnym.

3.2 Fitness

$$f = \begin{cases} C_{max}(I) & \text{jeżeli osobnik jest wykonywalny względem zasobów nieodnawialnych} \\ T + L^v(\mu) & \text{w.p.p} \end{cases} \tag{1}$$

$C_{max}(I)$ - czas wykonania prac z I,
T - suma maksymalnych czasów trwania prac z I
$L^v(\mu)$ - liczba przekroczonych zasobów nieodnawialnych

$$\begin{aligned} L_k^v(\mu) &= R_k^v - \textstyle\sum_{j=1}^{J} r_{j\mu(j)k}^v \\ L^v(\mu) &= \textstyle\sum_{k \in K^v, L_k^v(\mu) < 0} |L_k^v(\mu)| \end{aligned} \tag{2}$$

Taka definicja funkcji $fitness$ zapewnia, że osobniki wykonywalne względem zasobów nieodnawialnych będą lepsze niż te niewykonywalne.

3.3 Populacja inicjalna

Wielkość populacji jest równa POP. Proces generowania osobników:

1. losowy wybór $\mu(j) \in \mathbf{M_j}$ dla prac $j = 1, \ldots, J$
2. jeżeli osobnik jest niewykonywalny względem zasobów nieodnawialnych ($L_k^v(\mu) > 0$) wykonane zostaje przeszukiwanie lokalne: losowo wybierana jest praca $j \in J$, która ma więcej niż jeden możliwy tryb, następnie losowo wybierany jest tryb $m_j \neq \mu(j)$. Jeżeli nowy układ μ' jest lepszy lub równy wcześniejszemu układowi μ ($L_k^v(\mu') \leqslant L_k^v(\mu)$) to $\mu(j) = m_j$. Procedura jest powtarzana J razy lub jeżeli zostanie znaleziony osobnik wykonywalny względem zasobów nieodnawialnych

3.4 Crossover

Niech $I^M = (\lambda^M, \mu^M)$ będzie matką, a $I^F = (\lambda^F, \mu^F)$ ojcem. Po wykonaniu *crossover* powstanie córka $I^D = (\lambda^D, \mu^D)$ oraz syn $I^S = (\lambda^S, \mu^S)$.
Niech q_1, q_2 będą losowymi liczbami całkowitymi, takimi że $1 \leqslant q_1, q_2 \leqslant J$.

Prace :
Dla $i = 1, \ldots, q_1$: $j_i^D = j_i^M$.
Pozostałe prace na pozycjach $i = q_1 + 1, \ldots, J$ są dziedziczone od ojca pod warunkiem, że nie zostały odziedziczone wcześniej:
$j_i^D = j_k^F$, gdzie k jest najmniejszym indeksem, takim że $j_k^F \notin \{j_1^D, \ldots, j_{i-1}^D\}$.
Powyższa metoda gwarantuje, że pierwszeństwo prac zostanie zachowane.

Tryby :
Dla $i = 1, \ldots, q_2$: $\mu^D(j_i^D) = \mu^M(j_i^D)$.
Pozostałe tryby na pozycjach $i = q_2 + 1, \ldots, J$ są dziedziczone od ojca:
$\mu^D(j_i^D) = \mu^F(j_i^D)$
Syn I^S jest tworzony analogicznie, zamieniona jest kolejność dziedziczenia (najpierw od ojca, potem od matki).

Przykład :

$$\mathbf{I^M} = \begin{pmatrix} 2 & 4 & 1 & 6 & 3 & 5 \\ 2 & 2 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{I^F} = \begin{pmatrix} 1 & 3 & 2 & 5 & 4 & 6 \\ 1 & 2 & 1 & 1 & 2 & 2 \end{pmatrix}$$

$$\mathbf{I^D} = \begin{pmatrix} 2 & 4 & 1 & 3 & 5 & 6 \\ 2 & 2 & 1 & 1 & 1 & 2 \end{pmatrix}$$

$$\mathbf{I^S} = \begin{pmatrix} 1 & 3 & 2 & 4 & 6 & 5 \\ 1 & 2 & 1 & 2 & 1 & 1 \end{pmatrix}$$

3.5 Mutacja

Mutacja jest wykonywana na każdym osobniku, modyfikuje kolejność prac, a następnie trybów. Dla $i = 1, \ldots, J-1$ prace j_i i j_{i+1} są zamieniane miejscami z prawdopodobieństwem $p_{mutation}$ jeżeli po zmianie zachowane jest pierwszeństwo prac.
Następnie dla $i = 1, \ldots, J$ tryb $\mu(j_i)$ jest modyfikowany z prawdopodobieństwem $p_{mutation}$, nowy tryb jest losowo wybierany z $\mathbf{M_{j_i}}$.

Wartość $p_{mutation} = 0.05$ daje najlepsze wyniki.

3.6 Selekcja

Osobniki są sortowane po wartościach $fitness$, pozostaje POP osobników z najlepszą wartością $fitness$, pozostałe giną.

3.7 Przeszukiwanie lokalne

Czasy wykonywania prac reprezentowanych przez osobniki otrzymane z algorytmu genetycznego mogą zostać dodatkowo zmniejszone dzięki przeszukiwaniu lokalnemu. Osobnik $I = (\lambda, \mu)$ reprezentuje plan wykonywania prac s. Plan s może zostać polepszony poprzez zmianę trybu pracy j bez zmian w pozostałych pracach (multi-mode left shift). Dla każdej pracy $j_1, \ldots, j_J$ wykonywany jest multi-mode left shift, jeżeli jest to możliwe. Po procedurze przeszukiwania lokalnego otrzymujemy polepszony plan s', który odpowiada osobnikowi I'. Testy pokazały, że zamiana osobnika I na osobnika I' w populacji daje gorsze wyniki, gdyż powoduje utratę różnorodności genetycznej.

4 Porównanie MRCPSP [1] i JSP [2]

4.1 Generowanie nowej populacji

W JSP przy generowaniu nowej populacji wykorzystywana jest strategia elitarna. Część najlepszych osobników w populacji P_t jest kopiowana do populacji P_{t+1}. Następnie wykonywana jest operacja $crossover$, dwa osobniki są losowo wybierane z całej populacji P_t, dla każdego genu następuje rzut monetą z prawdopodobieństwem wyrzucenia orła równym p. Jeżeli zostanie wyrzucony orzeł gen jest dziedziczony z pierwszego rodzica, jeżeli reszka z drugiego.
Mutacja polega na losowym wygenerowaniu kilku nowych osobników.

4.2 Przeszukiwanie lokalne

Rozwiązanie JSP podobnie jak MRCPSP wykorzystuje algorytm genetyczny oraz przeszukiwanie lokalne. W JSP w procedurze przeszukiwania lokalnego zamieniane są miejscami zadania znajdujące się w jednym bloku, dzięki czemu zmniejsza się czas wykonania wszystkich zadań. W przeciwieństwie do MRCPSP polepszone rozwiązania są uwzględniane w populacji. W MRCPSP uwzględnianie polepszonych rozwiązań w populacji powodowało gorsze wyniki. Jest to spowodowane zmniejszeniem różnorodności genetycznej. Do nowej populacji trafia POP najlepszych osobników. W JSP populacja jest bardziej różnorodna dzięki mutacji polegającej na stworzeniu losowych osobników.

4.3 Wnioski

W obu algorytmach średni rozmiar POP populacji dawał najlepsze wyniki. W JSP jest to dwukrotna liczba wszystkich zadań. Przy zbyt dużej populacji limit czasu kończy się zanim zostanie znalezione optymalne rozwiązanie. Przy małej populacji różnorodność genów jest zbyt mała.
W JSP zbiór rozwiązań został ograniczony poprzez wprowadzenie maksymalnego dopuszczalnego czasu opóźnienia dla zadań. Zbiór danych bez tego parametru jest bardzo duży, gdyż zawiera również rozwiązania z długim czasem opóźnienia. Ograniczenie zbioru rozwiązań pozwala uzyskać lepsze wyniki.

Literatura

[1] S. Hartmann. *Project Scheduling with Multiple Modes: A Genetic Algorithm.*

[2] J.F. Goncalves et al. *A hybrid genetic algorithm for the job shop scheduling problem.*

[3] El-Ghazali Talbi, *Metaheuristics from design to implementation.*