ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: AUTOMATYKA z. 129 Nr kol. 1474

Marek ZASADA
Politechnika Szczecińska

ZASTOSOWANIE ALGORYTMÓW EWOLUCYJNYCH DO HARMONOGRAMOWANIA ZADAŃ W GNIEŹDZIE PRODUKCYJNYM

Streszczenie. Artykuł omawia procedury oraz wyniki zastosowania algorytmu ewolucyjnego do optymalizacji harmonogramu pracy gniazda produkcyjnego. Przedstawiono model matematyczny zadania harmonogramowania. Omówiono zastosowane operacje genetyczne: selekcję, krzyżowanie i mutację. Wyniki obliczeń opracowanego programu komputerowego dotyczą optymalizacji kolejności 15 zadań produkcyjnych realizowanych w gnieździe produkcyjnym złożonym z 10 stanowisk.

USE OF EVOLUTIONARY ALGORITHMS FOR SCHEDULING PRODUCTION TASKS IN WORKCELL

Summary. The paper discusses the procedures and the results application of evolutionary algorithm to the optimization of a workcell operation schedule. Presented is a mathematical model of a scheduling problem. Discussed are the genetic operations used: selection, crossing and mutation. The results of the computations of the computer program developed relate to the optimization of the sequence of 15 production tasks realized in a workcell composed of 10 workstands.

1. Wprowadzenie

Efektywność komputerowych metod stosowanych w technicznym przygotowaniu produkcji można poprawić wprowadzając do algorytmów wiedzę ekspercką z dziedziny rozpatrywanego problemu [4, 10, 11]. Dotyczy to w szczególności systemów wykorzystujących metody sztucznej inteligencji pomimo ich dużej uniwersalności i możliwości tworzenia oprogramowania typu szkieletowego (shells). Procesy technologiczne stanowiące podstawowy obiekt badań technologii maszyn cechuje [7, 10]:

1. Dyskretny sposób zmian charakterystyk przedmiotów obrabianych w kolejnych operacjach technologicznych (procesy technologiczne dyskretne).
2. Nieodwracalność skutków obróbki mechanicznej.
3. Szeregowy sposób łączenia poszczególnych etapów obróbki. Charakterystyka przedmiotu po zakończeniu danego etapu jest charakterystyką wejściową etapu następnego.

4. Wpływ charakterystyk, w szczególności dokładności wymiarowo-kształtowej przedmiotu po danej operacji technologicznej, na wyniki operacji następnej (zasada wzrastającej dokładności wymiarowo-kształtowej).

Struktura procesu i poszczególnych operacji technologicznych tworzona jest na podstawie pewnej liczby reguł opisujących tzw. strukturę fazowo-stopniową procesu, reguł wyboru baz obróbkowych, zasady stałości baz obróbkowych itp. Znaczny jednak obszar pracy technologa opiera się na wiedzy heurystycznej, wynikającej z jego doświadczenia przy projektowaniu podobnych procesów technologicznych oraz ich wdrażaniu w konkretnych warunkach warsztatowych. Na ogół występuje potrzeba analizy wielu wariantów dopuszczalnych procesów, przy czym wybór wariantu optymalnego jest trudny ze względu na liczne kryteria optymalizacji, ograniczenia techniczne oraz potrzebę ocen eksperckich, często o charakterze rozmytym. Istotnym ułatwieniem projektowania procesów technologicznych jest wykorzystanie metod opartych na klasyfikacji przedmiotów według podobieństwa konstrukcyjno-technologicznego. Powstaje wówczas możliwość opracowania ujednoliconych struktur procesów dla określonych klas części takich, jak: wały, korpusy, koła zębate, dźwignie itp. Metody typizacji procesów technologicznych oraz obróbki grupowej (w zakresie operacji technologicznej), chociaż znane od lat, mogą być skutecznie wykorzystane dopiero przy szerokim wsparciu techniki komputerowej oraz systemów z zakresu CAD/CAM. Znaczenie tych metod w technologicznym i organizacyjnym przygotowaniu produkcji wzrasta w związku z rozwojem zautomatyzowanych i zintegrowanych systemów produkcyjnych. Jednym z istotnych warunków ekonomicznego wykorzystania takich systemów jest posiadanie oprogramowania umożliwiającego szeregowanie zadań przyjętych do realizacji [3, 4, 8, 9]. Optymalny harmonogram pracy gniazda produkcyjnego umożliwia maksymalne wykorzystanie dysponowanego czasu pracy poszczególnych stanowisk, czyli wykonanie zadania produkcyjnego w najkrótszym czasie.

2. Matematyczny model zadania harmonogramowania

Gniazdo produkcyjne jest zbiorem stanowisk obróbkowych (np. obrabiarek CNC) uzupełnionych stanowiskami nieobróbkowymi (np. kontroli jakości, obróbki cieplnej, mycia). Zbiór stanowisk w gnieździe produkcyjnym o liczebności s można zapisać jako:

$$S=\{S_1, S_2, \ldots, S_i, \ldots, S_s\}.$$

Mając na uwadze specyfikę procesów technologicznych zadanie harmonogramowania sformułowano następująco.

Dany jest z elementowy zbiór zadań produkcyjnych:

$$Z=\{Z_1, Z_2, \ldots, Z_i, \ldots, Z_z\}.$$

Zadanie Z_i dotyczy serii takich samych przedmiotów o liczebności n_i ($n_i \geq 1$) wykonywanych według tej samej marszruty technologicznej. Każda marszruta stanowi ciąg operacji

technologicznych realizowanych na określonych stanowiskach według ustalonej przez technologa kolejności. Marszrutę i można zapisać następująco:

$$O_i=(O_{i1}, O_{i2}, \ldots, O_{ij}, \ldots, O_{is}).$$

Rodzaj nawiasów podkreśla istotność kolejności operacji technologicznych w marszrucie. Przyjęto, że gniazdo produkcyjne pracuje według zasad typizacji procesów technologicznych, w związku z czym istnieje ujednolicona marszruta (proces technologiczny typowy), opracowana dla przedstawiciela danego typu technologicznego. Ze względu na różne wymiary przedmiotów czasy trwania poszczególnych operacji w marszrutach indywidualnych są różne. Przyjęcie czasu trwania pewnej operacji równego zero jest równoważne pominięciu jej w marszrucie. Z punktu widzenia przyporządkowania operacji technologicznych do stanowisk można badać trzy tryby pracy gniazd produkcyjnych:

a) *sekwencyjny*, gdy każda operacja jest realizowana na ściśle określonym stanowisku,

b) *dowolny*, gdy każda operacja może być wykonywana na każdym stanowisku,

c) *mieszany*, gdy dla niektórych operacji przewidziano alternatywne stanowiska, natomiast dla innych operacji obowiązuje ścisłe przypisanie do stanowisk.

Przyjmując sekwencyjny tryb pracy gniazda, problem harmonogramowania można sformułować jako określenie kolejności zadań na poszczególnych stanowiskach tak, aby optymalizować wybraną funkcję celu. Wyznaczona kolejność zadań jest podstawą obliczenia terminów rozpoczęcia P_{ji} oraz zakończenia K_{ji} poszczególnych operacji (indeks i) w każdym zadaniu (indeks j). Z wielu możliwych funkcji celu [6] w dalszej części pracy przyjęto czas zakończenia wszystkich zadań, który powinien być minimalizowany:

$$K_{max}= max\ K_{js} \rightarrow min$$

gdzie: K_{js} - termin zakończenia ostatniej operacji w zadaniu Z_j $(j=1, \ldots, z)$.

Ponieważ standardowe algorytmy ewolucyjne rozwiązują zadania maksymalizacji, dokonano przekształcenia przyjętej funkcji celu do postaci nazywanej w nomenklaturze tych algorytmów funkcją przystosowania F_p (fitness function):

$$F_p = C - K_{max} \rightarrow max,$$

gdzie: C – stała zapewniająca warunek $F_p>0$.

Istotnym uproszczeniem, jakie przyjęto przy opracowaniu programu do harmonogramowania, jest przyjęcie jednakowej kolejności zadań na wszystkich stanowiskach. Założenie to ułatwia opracowanie aplikacji algorytmu ewolucyjnego, może jednak pogorszyć wskaźniki harmonogramu, zwłaszcza przy dużej liczbie stanowisk i bardzo zróżnicowanych czasach trwania operacji technologicznych. W kolejnej wersji programu uproszczenie to zostanie wyeliminowane. Podstawą reprezentacji problemu w algorytmie ewolucyjnym jest więc kolejność zadań zapisywana jako ciąg liczb oznaczających ich numery (permutacja numerów zadań). Obszarem, w którym poszukuje się rozwiązania, jest zatem zbiór permutacji numerów zadań:

$$\{(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_z): \sigma_i \neq \sigma_j\},$$

gdzie: $\sigma_i \in <1, 2, 3, \ldots z>$ - numery zadań.

Liczba elementów zbioru permutacji rośnie wraz ze wzrostem liczby zadań zgodnie z funkcją *silnia*, której dobre oszacowanie daje znany wzór Stirlinga:

$$z! \approx \sqrt{2\pi z}\, z^z e^{-z},$$

gdzie: e - podstawa logarytmów naturalnych.

Jest to wzrost szybszy niż wzrost funkcji wykładniczej. Można wykazać, że dla dowolnej liczby $a>1$ należącej do zbioru liczb naturalnych $(a \in N)$, począwszy od pewnego $z \in N$, zachodzi nierówność:

$$z! > a^z.$$

Doświadczenie numeryczne pokazuje, że ze względu na czas obliczeń metoda systematycznego przeszukiwania staje się nierealna już od liczby zadań $z=13$ ($13! \approx 6.2 * 10^9$). Postawione zadanie należy do klasy NP-trudnych, a do jego rozwiązania wykorzystuje się najczęściej metody heurystyczne oraz ostatnio metody oparte na sztucznej inteligencji. W każdej z tych metod dąży się do minimalizacji liczby działań matematycznych, co dzieje się na ogół kosztem zmniejszenia dokładności uzyskiwanych wyników.

3. Algorytm ewolucyjny i wybrane szczegóły operacji genetycznych

Podstawowym elementem pokolenia w algorytmie ewolucyjnym, nazywanym osobnikiem lub chromosomem, będzie ciąg numerów zadań, ustalający kolejność realizacji zadań jednakową na każdym stanowisku. Zbiór chromosomów, tworzący pokolenie, będzie ewoluował zgodnie z zasadami algorytmu przedstawionego na poniższym wydruku:

1. Wczytanie danych.
2. Inicjalizacja losowa populacji:
   - ustawienie licznika pokoleń: *t:=1,*
   - zapis tablicy pokolenia nr 1 - tablica *Pop1[..].*
3. Obliczenie funkcji przystosowania, statystyk pokolenia nr 1, generowanie raportu, zapis wyników na dysk.
4. ***REPEAT*** (powtarzaj)
   - inkrementacja licznika pokoleń: *t:=t+1*
   - wykonanie na bieżącym pokoleniu operacji genetycznych:
     - selekcja, krzyżowanie, mutacja,
     - zapis pokolenia nowego w tablicy *Pop2[..],*
   - obliczenie funkcji przystosowania, statystyk pokolenia, zapamiętanie najlepszego chromosomu (permutacji zadań), generowanie raportu, zapis wyników na dysk,
   - wymiana pokoleń: *Pop1[..]:=Pop2[..].*

   UNTIL (warunek zakończenia iteracji)
5. Obliczenie tablicy czasów zakończenia każdej operacji T[i,j] na podstawie zapisanej w najlepszym chromosomie kolejności zadań.
6. Wydruk wyników.

Obliczenie przyjętej funkcji przystosowania jest dość uciążliwe [2] i wymaga opracowania odrębnej procedury. Podstawowe dane wejściowe do procedury to liczba zadań *nz*, liczba maszyn *nm*, tablica czasów trwania operacji *Z[..]* oraz aktualna postać permutacji *P[..]*. W dwóch pętlach, przebiegających wszystkie zadania oraz wszystkie stanowiska, analizowany jest cały harmonogram i obliczany jest czas zakończenia wszystkich zadań, który następnie służy do obliczenia funkcji przystosowania. Poniższy wydruk przedstawia procedurę obliczania czasu zakończenia wszystkich zadań, opracowaną w postaci funkcji *FP(..)* w języku Pascal.

```
function FP(nz,nm:Integer; Z:tabz; P:tabper):real;
{obliczenie czasu zakonczenia wszystkich zadan}
var
i,ip,nrz,im: Integer;
t:tabczas;                    {tablica czasu pracy stanowisk}
begin
 for i:=1 to nm do t[i]:=0; {zerowanie bieżącego czasu pracy stanowisk}
 for ip:=1 to nz do          {powtarzaj zmieniając zadania}
  begin
   nrz:=P[ip];               {numer zadania}
   for im:=1 to nm do        {powtarzaj zmieniając stanowiska}
    if (im=1) or ((im>1) and (t[im]>t[im-1]))
      then t[im]:=t[im]+Z[im,nrz]
      else t[im]:=t[im-1]+Z[im,nrz];
  end;
 FP:=maxt(t,nm)
end;
```

W programie przyjęto operację selekcji opartą na zmodyfikowanej metodzie ruletki. Modyfikacja polega na wprowadzeniu progu nie pozwalającego na rozprzestrzenianie się osobników (chromosomów) o bardzo małej wartości funkcji przystosowania. Eliminowanie bardzo "słabych" osobników odbywa się poprzez porównanie ułamka reprezentującego udział w całej populacji ich funkcji przystosowania z wartością *ps/npop*, gdzie:

ps - ustalona doświadczalnie wartość progu selekcji,

npop - liczebność populacji.

Ściśle mówiąc zastosowany próg nie pozbawia całkowicie szans powielenia słabego osobnika. Szansa taka jest jednak bardzo mała (musi wystąpić przypadek równości pozycji słabego osobnika w starym pokoleniu z bieżącą pozycją w generowanym nowym pokoleniu). Zapis procedury selekcji w języku Pascal przedstawia poniższy wydruk:

```
procedure selekcja(nz,npop:Integer;ps:Real;Pop1:tabpop; cc4:tabcc4;
                   var Pop2:tabpop);
{nz - liczba zadan, npop - liczebnosc populacji, ps - prog selekcji,
 Pop1 - tablica populacji starej, cc4 - tabl.ze statystykami pokolenia,
 Pop2 - tablica populacji nowej}
var
 i,j,k:Integer;
 r:Real;
begin
 for i:=1 to npop do
  begin
   j:=0;
   r:=random;
```

```
   repeat
    j:=j+1;
   until r<=cc4[j,4]; {powtarzaj az wyrazenie stanie sie prawda}
   if cc4[j,3]>(ps/npop)
    then
     for k:=1 to nz do Pop2[i,k]:=Popl[j,k]
    else
     for k:=1 to nz do Pop2[i,k]:=Popl[i,k];
  end;
end;
```

Operacja krzyżowania ma na celu połączenie materiału genetycznego przynajmniej dwóch osobników w nadziei uzyskania potomka lepiej przystosowanego niż para rodzicielska. Przy kodowaniu zadania w liczbach zmiennoprzecinkowych operacja ta nie jest tak prosta i dobrze uzasadniona teoretycznie jak przy kodowaniu binarnym (koncepcja cegiełek w algorytmach genetycznych) [1, 5]. W opracowanym programie przyjęto procedurę, którą w skrócie można przedstawić następująco: dla danego osobnika dobierany jest losowo partner, a następnie, również losowo, określana jest pozycja podziału chromosomu. Potomek uzyskuje od pierwszego rodzica początkową część chromosomu, natomiast część końcowa (tzn. od pozycji podziału do końca chromosomu) dobierana jest z rodzica drugiego. Część końcowa chromosomu tworzona jest na zasadzie uzupełniania brakujących elementów w prawidłowo zbudowanej permutacji. Od drugiego rodzica potomek dziedziczy uzupełniane elementy oraz kolejność ich ustawienia. Poniższy przykład pokazuje możliwy wynik krzyżowania dwóch chromosomów 9-genowych:

```
Chromosom nr 1:   7  9  8  1  |  4  2  3  5  6 (pozycja cięcia 4)
Chromosom nr 2:   3  7  4  9  2  1  8  6  5
Potomek:          7  9  8  1  3  4  2  6  5
```

Zaprogramowano również kontrolę poprawy funkcji przystosowania po operacji krzyżowania i możliwość wycofania się z tej operacji w przypadku jej pogorszenia. Omówioną technikę krzyżowania przedstawia poniższa procedura zapisana w języku Pascal:

```
procedure krzyzowanie(nz,nm,npop:Integer; var Pop2:tabpop);
var
 i,il,j,k,kl,k2,k3,kon:Integer;
 fc1,fc2:Real;
 P2:tabper;
begin
 For i:=1 to npop do
  begin
   for j:=1 to nz do P[j]:=Pop2[i,j];
   fc1:=FP(nz,nm,Z,P);
   for j:=1 to nz do P2[j]:=Pop2[i,j]; {chwilowe zapamietanie osobnika}
   il:=Random(npop)+1;   {losowanie partnera do krzyzowania}
   kl:=Random(nz-2)+1;   {losowanie pozycji ciecia chromosomu}
   for k:=kl+1 to nz do  {petla po 'ogonie' chromosomu}
    begin
     for k2:=1 to nz do  {petla po genach partnera}
      begin
       kon:=0;
       for k3:=1 to k-1 do          {petla po 'glowie' chromosomu}
        if Pop2[i,k3]=Pop2[il,k2] then kon:=1;
```

```
        if kon=0 then
         begin
          Pop2[i,k]:= Pop2[il,k2];
          k2:=nz;
         end;
       end; {koniec petli po genach partnera}
     end;   {koniec petli po 'ogonie' chromosomu}
    for j:=1 to nz do P[j]:=Pop2[i,j];
    fc2:=FP(nz,nm,Z,P);
    if fc2>fcl       {operacja krzyzowania nie przyniosla poprawy}
     then for j:=1 to nz do Pop2[i,j]:=P2[j]; {przywrocenie chromosomu}
  end;      {koniec petli po osobnikach w Pop2}
end;      {koniec procedury krzyzowania}
```

Standardowa operacja mutacji w algorytmach genetycznych, przy binarnym kodowaniu zadania, polega na losowaniu chromosomu oraz pozycji w chromosomie, a następnie zmianie wartości stojącego tam bitu na wartość przeciwną. Wzorując się na obserwacjach z przyrody ożywionej mutacji poddaje się na ogół bardzo małą liczbę chromosomów. W algorytmach ewolucyjnych, przy kodowaniu zadania w liczbach zmiennoprzecinkowych, procedurę mutacji opracowuje się najczęściej indywidualnie dla danego typu zadania. W omawianym programie przyjęto operację mutacji, polegającą na losowaniu dwóch pozycji w chromosomie i zamianie stojących tam wartości reprezentujących geny. Możliwą operację mutacji przedstawia następujący przykład:

```
Chromosom pierwotny:   3  9  8  1  4  2  7  5  6
Chromosom po mutacji:  3  5  8  1  4  2  7  9  6
```

W praktyce okazało się, że w tej operacji również ma znaczenie bieżąca kontrola poprawy funkcji przystosowania.

W procedurach algorytmu realizujących operacje genetyczne oprócz mechanizmów kontrolujących poprawę funkcji przystosowania wprowadzono współczynniki prawdopodobieństwa, decydujące o częstotliwości wystąpienia tych operacji, a nawet umożliwiające ich całkowite wyłączenie (współczynnik równy zero). Ustalenia wartości wszystkich parametrów pracy algorytmu ewolucyjnego, w tym liczebności populacji oraz liczby pokoleń, dokonano na podstawie serii próbnych obliczeń. Celem „strojenia" pracy algorytmu jest uzyskanie odpowiedniej jego zbieżności. Zbyt szybka zbieżność (tzw. przedwczesna) prowadzi z reguły do "utknięcia" w optimum lokalnym i pojawienia się cykliczności wyników w kolejnych iteracjach. Zbyt wolna zbieżność wymaga generowania dużej liczby pokoleń wydłużając czas pracy programu.

4. Testy i ocena pracy programu

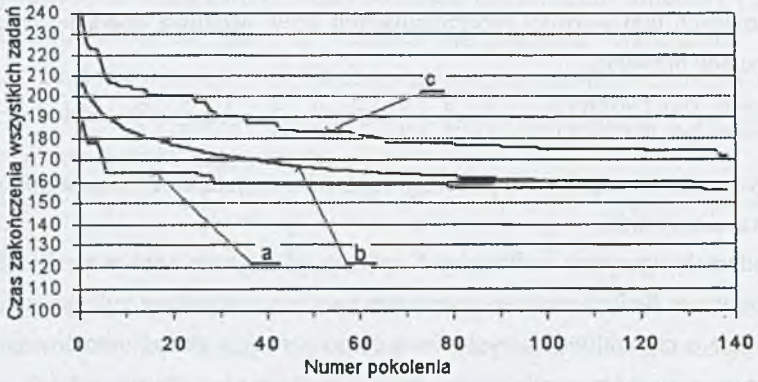
Do oceny pracy programu przygotowano szereg testów o różnej liczebności zbiorów stanowisk i zadań. Dalej pokazano przykład optymalizacji harmonogramu pracy gniazda produkcyjnego złożonego z 10 stanowisk, na których ma być uruchomionych 15 zadań, opracowanych według zasad typizacji procesów technologicznych (przyjęto n_s=1). W tablicy 1 ze-

stawiono czasy operacji technologicznych stanowiące dane wejściowe do programu. Na rys. 1 pokazano wykres zmian statystyk w kolejnych pokoleniach, opracowany na podstawie wyników nagrywanych w każdej iteracji do pliku dyskowego.

Tablica 1

Czasy operacji technologicznych [min]

Nr stanowiska	Numer zadania														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	4	4	9	11	6	4	1	7	10	5	10	2	10	2
2	2	5	2	7	4	4	7	4	7	4	6	1	10	3	10
3	3	7	6	9	7	10	12	2	7	3	10	7	1	2	1
4	5	4	7	10	1	4	4	10	7	15	9	15	2	9	7
5	8	3	8	4	5	10	6	7	7	9	2	11	5	11	3
6	3	8	9	3	7	11	8	13	2	12	7	3	3	13	2
7	10	9	4	7	11	2	5	6	10	3	12	1	7	5	6
8	15	12	7	3	1	4	8	10	7	8	2	9	1	4	2
9	5	5	10	3	9	15	2	7	15	6	4	4	8	8	4
10	1	4	9	11	8	1	5	12	3	4	1	8	15	10	1



Rys.1. Statystyki kolejnych pokoleń algorytmu ewolucyjnego: a, b, c - wartości minimalna, średnia i maksymalna czasu realizacji harmonogramu

Fig.1. Statistics of successive generations of evolutionary algorithm: a, b, c, - minimum, mean and maximum values of schedule realization time

W przedstawionym przykładzie optymalną kolejność zadań przedstawia następujący ciąg ich numerów: 1-3-13-8-9-2-14-5-7-10-4-6-12-15-11. Odpowiadający tej kolejności czas realizacji harmonogramu wynosi 156 minut. Graficzną reprezentacją wyników optymalizacji harmonogramu może być wykres Gantta, który przedstawiono na rys, 2.

W zależności od liczby zadań i ustawień parametrów algorytmu akceptowalne wyniki optymalizacji uzyskiwano po liczbie pokoleń mieszczącej się w granicach 20-150. Dla małej liczby zadań, możliwej do weryfikacji metodą systematycznego przeszukiwania, uzyskiwano bez problemów optimum globalne. Dla dużej liczby zadań wskazane jest kilkakrotne powtórzenie obliczeń z ewentualną zmianą parametrów algorytmu ewolucyjnego.

Rys.2. Wykres Gantta przedstawiający optymalny harmonogram pracy gniazda obróbkowego uzyskany na podstawie wyników obliczeń algorytmu ewolucyjnego
Fig.2. A Gantt chart which presents an optimum schedule of workcell operation, obtained based on the results of computations of evolutionary algorithm

5. Wnioski

1. Wyniki obliczeń potwierdziły zasadność wykorzystania algorytmu ewolucyjnego do optymalizacji zadań harmonogramowania. Algorytm ten można łatwiej przystosować do specyfiki rozwiązywanego problemu niż podstawowy algorytm genetyczny. Przedstawiony przykład (10 stanowisk, 15 zadań) jest wymiarowo zbliżony do typowych zadań testowych, np. znanego zadania Fishera-Thompsona MT-10x10. Funkcjonowanie opracowanego programu ze względu na czas obliczeń i powtarzalność wyników było praktycznie sprawdzane dla przynajmniej dwukrotnie większej liczby zadań i liczby stanowisk.

2. Optymalizacja harmonogramu wg kryterium minimalnego czasu trwania wszystkich zadań minimalizuje jednocześnie czas oczekiwania maszyn na kolejne zadania. Czas ten można z kolei wykorzystać do obniżenia technologicznych parametrów skrawania i tym samym wydłużenia okresu trwałości narzędzia oraz obniżenia składnika kosztów narzędziowych danej operacji [8].

3. Optymalizacja harmonogramu w ogólnym przypadku jest zadaniem NP-trudnym. Tworzenie optymalnego harmonogramu pracy gniazda produkcyjnego sprowadzono do przeszukiwania zbioru permutacji, a więc zadania optymalizacji kombinatorycznej analogicznego do znanego „problemu komiwojażera" (przeszukiwanie zbioru permutacji numerów miast w celu znalezienia najkrótszej drogi).

4. Rozwój prac nad wykorzystaniem algorytmów ewolucyjnych w harmonogramowaniu zmierzał będzie do uwzględnienia transportu pomiędzy stanowiskami, obsługi przez robota

przemysłowego, możliwości dynamicznego projektowania harmonogramu w zmiennych warunkach produkcyjnych oraz usunięcia ograniczenia dotyczącego stałej kolejności zadań na wszystkich stanowiskach.

LITERATURA

1. Bäck T., Hammel U., Schwefel H.: „Evolutionary Computation: Comments on the History and Current State", IEEE Transactions on Evolutionary Computation, Vol 1, No 1, April 1997, pp. 3 - 12.
2. Knosala R., Wal T.: Budowa funkcji przystosowania algorytmu genetycznego w procesie harmonogramowania pracy elastycznego gniazda produkcyjnego. Materiały konferencji „Komputerowo Zintegrowane Zarządzanie", Zakopane 1998.
3. Knosala R., Wal T.: Sposób oceny ciągu kodowego w harmonogramowaniu produkcji za pomocą algorytmów genetycznych. Materiały XIII Krajowej Konferencji Automatyki, Opole 1999.
4. Kampa A., Knosala R.: Harmonogramowanie produkcji w gnieździe wytwórczym z zastosowaniem systemu doradczego. Materiały konferencji „Komputerowo Zintegrowane Zarządzanie", Zakopane 1998.
5. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne, WNT, Warszawa 1996.
6. Pawlak M.: Algorytmy ewolucyjne jako narzędzie harmonogramowania produkcji. PWN, Warszawa 1999.
7. Szulc S.: Struktura technologicznych procesów budowy maszyn. Archiwum budowy maszyn, 9 (1962) 1.
8. Szwabowski J., Zasada M.: Use of genetic algorithms in parametric multiobjective optimization of multi-tool operations with simultaneously active tools. Advances in technology of the Machines and Equipment. Kwartalnik. Polska Akademia Nauk, Vol. 23, Nr 2, 1999, (71-97).
9. Witkowski T.: Metoda optymalizacji harmonogramów pracy maszyn. Mechanik nr 8-9/93 ss. 305-308.
10. Zasada M., Żaboklicki S: Technological knowledge representation in expert systems on example of the selection of technological process structure. Advances in technology of the Machines and Equipment. Kwartalnik. Polska Akademia Nauk, VOL. 20, Nr 2, 1996 (99-112).
11. Zasada M., Chmielewski K: Zastosowanie metod sztucznej inteligencji w projektowaniu procesów technologicznych obróbki skrawaniem. Materiały konferencji: „Współczesne problemy w budowie i eksploatacji maszyn" Politechnika Szczecińska, Szczecin 1996 (73-82).

Recenzent: Prof.dr hab.inż. A.Niederliński

Abstract

1. Problem formulation and mathematical model

A condition of an economical utilization of production systems is the availability of a software which would allow jobs taken of for realization to be scheduled. An optimum work

schedule of e.g. a workcell enables us to utilize maximally the available time of the machinery operation, i.e. to perform a production task in the shortest time. The purpose of scheduling is to find the sequences of jobs realized on particular workstands, and to determine the times of commencing P_j and completing K_j of these jobs so that the adopted objective function reach its optimum. The input data include a set of production tasks, **Z**, a set of technological routes $\{O_i\}$, the times of particular technological operations $\{T_i\}$, a set of workstands in a workcell, **S**, and the machinery operation mode.

Determination of the sequence of production tasks boils down to seeking an optimal permutation of their numbers. This is a problem from the scope of the combinatory optimization with a degree of complexity: NP–difficult. The systematic search method fails when the number of tasks, $z > 13$ ($13! \approx 6.2 * 10^9$). Adopted as an objective function was the time of completing all the tasks which was subject to minimization: $K_{max} = max\ K_{js} \rightarrow min$. The fitness function in evolutionary algorithms which was subject to maximization is given by $F_p = C - K_{max} \rightarrow max$, where C is a constant that is assured of $F_p > 0$. A graphical presentation was made making use of Gantt charts.

2. Method of solving

Used was an evolutionary algorithm, classified among the artificial intelligence methods, the scheme of which is presented by the following printout:

```
1. Data reading in.
2. Random initialization of first population, Pop1[..].
3. Calculation of fitness function and statistics of Generation No. 1.
REPEAT
      - performing genetic operations of selection, crossing and mutation
        on the current generation,
      - record of new generation in Table Pop2[..],
      - calculation of fitness function and statistics of new generation,
      - generation replacement: Pop1[..]:=Pop2[..].
UNTIL (iteration termination condition)
5. Printout of results.
```

The selection operation adopted in the program is based on a modified roulette method. The modification consist in introducing a threshold which prevents individuals (chromosomes) of a very low value of fitness function from spreading. The crossing operation can by briefly presented as follows: for given individual a partner is selected randomly and, next – also randomly – the position of the chromosome division is determined. The descendant acquires the initial part of chromosome from its first parent while the end part is selected from the next parent on the principle of completing missing elements. The following example shows a possible result of the operation of crossing 9 – element chromosomes:

```
Chromosome No. 1:  7 9 8 1 | 4 2 3 5 6   (cutting at position 4)
Chromosome No. 2:  3 7 4 9 2 1 8 6 5
Descendant:        7 9 8 1 3 4 2 6 5
```

The mutation consisted in sampling two positions in the chromosome and replacing the values which stood there and which represented the genes. The following example presents a possible result of the mutation:

```
Primary chromosome:          7 9 8 1 3 4 2 6 5
Chromosome after mutation:   7 6 8 1 3 4 2 9 5
```

Each procedure which realized genetic operation was provided with a programmed mechanism for checking the correctness of the fitness function to allow the algorithm converging rate to be controlled. Moreover to each of the genetic operations is assigned a probability factor that determines the frequency of use (or a total exclusion) of these operations. The values of all the parameters of the evolutionary algorithm operation, including the population size, the number of generations and the probability factors, were determined based on trial computations carried out so as to achieve an adequate convergence of the algorithm.

3. Program test and Conclusions

In order to evaluate the program operation, a series of tests characterized by various size of the sets of machines and tasks were prepared. Figure 1 shows the course of the optimization of operation of a workcell composed of 10 workstands. The optimization dealt with 15 workparts (production tasks) whose operation times are compiled in Table1. Figure 2 shows a Gantt chart that corresponds to the optimum sequence of tasks started in the workcell. Conclusions drawn from the computations realized are presented in the following points:

1. Optimization of workcell operation schedule is an NP-difficult problem, reduceable to the traveling salesman problem with an adequately prepared objective function.
2. The criterion of the minimum duration of all the tasks minimizes simultaneously the waiting time of machines for successive tasks. A known waiting time can be utilized for reducing technological cutting parameters, which prolongs the tool life and decreases the tool cost component in given operation.
3. The results of the computations confirmed the rightness of the application of evolutionary algorithms to the scheduling problem optimization. These algorithms can be adapted more easily than the basic genetic algorithm to the specificity of the problem solved.
4. The next works should employ workcell models which take into account the time of servicing machine tools by an industrial robot, and allow for the possibility of a quick modification of the schedule, depending on the inflow of new orders, the priorities of these orders, or the disturbances in the machinery operation for technical reasons.