Harmonogramowanie zadań wielomaszynowych z przezbrojeniami

Michał Zimniak, Norbert Januszek

Uniwersytet Wrocławski

zimniak.michal@gmail.com traspie@wp.pl

16 grudnia 2016

Wstęp

Będziemy rozpatrywać problem szeregowania zadań wielomaszynowych z przezbrojeniami maszyn. Chcemy zminimalizować iloczyn liczby użytych maszyn i długość wykonywania wszystkich zadań.

Specyfikacja problemu

Dane wejściowe

- Zbiór zadań $\mathcal{J} = \{1, 2, \ldots, n\}$
  - Zadanie $i \in \mathcal{J}$ wymaga jednocześnie $size_i$ maszyn przez $p_i > 0$ jednostek czasu.
- Zbiór maszyn $\mathcal{M} = \{1, 2, \ldots, m\}$
  - Maszyny $l, k \in \mathcal{M}$ nazywamy sąsiednimi, jeżeli $k = l + 1$ lub $k = l - 1$.
  - Przez $s_{i,j}$ oznaczamy czas przezbrojenia pomiędzy zadaniem i, a zadaniem j.

Specyfikacja problemu

Założenia

- Żadna maszyna nie może wykonywać więcej niż jedno zadanie w danym momencie.
- Wykonywanie zadania nie może być przerwane.
- Każde zadanie jest wykonywane na wymaganej liczbie sąsiednich maszyn.
- Pomiędzy kolejno wykonywanymi zadaniami należy wykonać przezbrojenie maszyny.
- Zakładamy, że liczba maszyn $m \geqslant \max\{size_i : i \in \mathcal{J}\}$, czyli jesteśmy w stanie wykonać każde zadanie.

Specyfikacja problemu

Rozwiązanie

- Należy dla każdego zadania wyznaczyć podzbiór maszyn oraz momentów rozpoczęcia jego wykonywania spełniając wymienione ograniczenia, tak żeby zminimalizować iloczyn liczby użytych maszyn i długość wykonywania wszystkich zadań.

Rozwiązanie może być reprezentowane przez parę $\Theta = (\mathcal{Q}, \mathcal{S})$ taką, że:

- $\mathcal{Q} = (\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_n)$, gdzie $\mathcal{Q}_i \subseteq \mathcal{M}$ to zbiór maszyn na których będzie wykonywane zadanie $i \in \mathcal{J}$.
- $\mathcal{S} = (\mathcal{S}_{1,1}, \ldots, \mathcal{S}_{1,m}, \mathcal{S}_{2,1}, \ldots, \mathcal{S}_{2,m}, \ldots, \mathcal{S}_{n,1}, \ldots, \mathcal{S}_{n,m})$, gdzie $\mathcal{S}_{i,j}$ jest momentem rozpoczęcia wykonywania zadania i na maszynie j. Jeżeli $j \notin \mathcal{Q}_i$ to $\mathcal{S}_{i,j} = -\infty$.

Specyfikacja problemu

Funkcja celu

- $F(\Theta) = C_{max}(\Theta) \cdot M_{max}(\Theta)$
- $C_{max} = max\{\mathcal{S}_{i,j} + p_i : i \in \mathcal{J}, j \in \mathcal{M}\}$ jest momentem zakończenia wykonywania wszystkich zadań.
- $M_{max} = max\{j : j \in \mathcal{Q}_i, i \in \mathcal{J}\}$ jest maksymalnym numerem maszyny spośród wszystkich przydzielonych do wykonywania zadań.

Oznaczmy przez Ω zbiór wszystkich rozwiązań. Problem harmonogramowania zadań polega na wyznaczeniu rozwiązania $\Theta^* \in \Omega$ takiego, że:

$$F(\Theta^*) = min\{F(\Theta) : \Theta \in \Omega\}.$$

Charakterystyka problemu

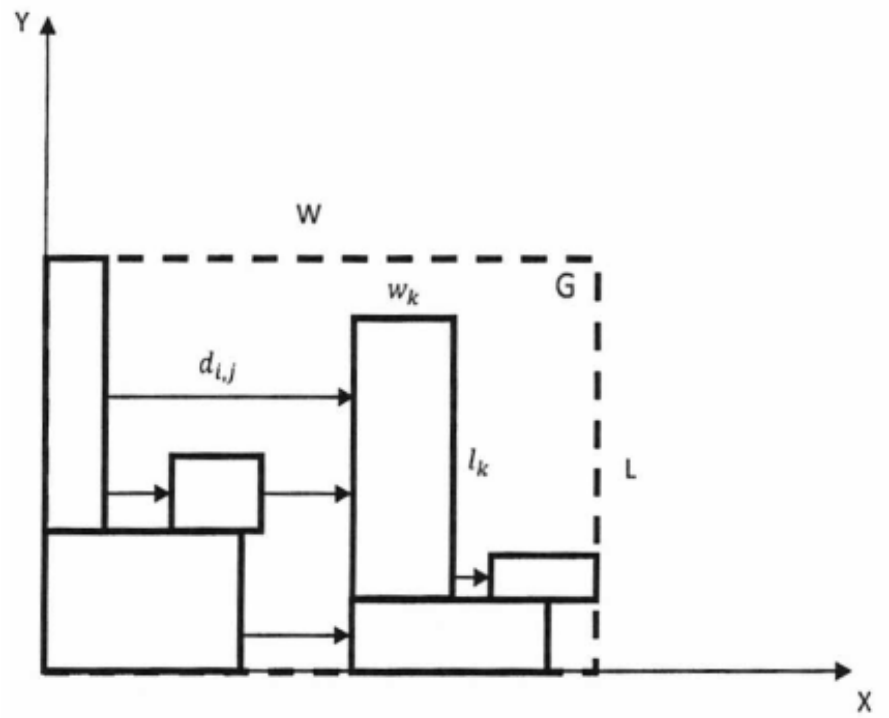
- Okazuje się, że problem jest równoważny pewnemu uogólnieniu dwuwymiarowego problemu pakowania.
- Są to problemy NP-trudne. Algorytmy wyznaczania rozwiązania optymalnego mają wykładniczą złożoność, dlatego mogą być stosowane jedynie do rozwiązywania przykładów o niewielkich rozmiarach.
- Algorytmy przybliżone często dają bardzo zadowalające wyniki różniące się od najlepszych rozwiązań o mniej niż 15%.
- Oczywiste dolne oszacowanie rozwiązania optymalnego problemu pakowania może być sumą pól wszystkich prostokątów.

Dwuwymiarowy problem pakowania

Należy tak rozmieścić prostokąty, aby zajmowany przez nie obszar G miał minimalne pole powierzchni. W rozważanym wariancie prostokąty, które stykają się bokami należy odsunąć od siebie na pewną odległość.

Specyfikacja

- $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ - zbiór prostokątów
- (l_k, w_k) - wysokość i szerokość prostokąta r_k
- $d_{i,j}$ - odległość o jaką należy odsunąć prostokąt r_i od r_j, gdy występują bezpośrednio obok siebie



Algorytm pakowania

Niech $\alpha = (r_1, r_2, \ldots, r_n)$ będzie pewnym ciągiem prostokątów. Rozpatrujemy prostokąty zgodnie z kolejnością ich występowania w α. Zbiór P_{i-1} zawiera pozycje, na których można umieścić rozpatrywany prostokąt r_i.

$G \leftarrow \emptyset, P_0 \leftarrow \{(0,0)\}$
for $i \leftarrow 1, n$ **do**
  $O \leftarrow \emptyset$
  for z in P_{i-1} **do** $O \leftarrow O \cup \{insert(G, z, r_i)\}$
  $G \leftarrow$ wybrany na podstawie ustalonej strategii obszar z O powstały po wstawieniu r_i na pozycję $z = (x, y)$.
  $P_i \leftarrow P_{i-1} \setminus \{z\} \cup \{(x + d_{r_c,r_i} \cdot \delta, y), (x + d_{r_c,r_i} \cdot \delta, y + l_i)\}$

$\delta \in \{0, 1\}$ i przyjmuje wartość 1, jeżeli wstawiany prostokąt r_i został przesunięty o wielkość d_{r_c,r_i}, a pozycja z została utworzona w wyniku wstawienia prostokąta r_c.

Strategie pakowania

1. *minimalnego obszaru* - wybierana jest pozycja, powodująca minimalne powiększenie bieżącego obszaru.
2. *pole-wymiary* - obliczana jest wartość współczynnika na podstawie pola powierzchni tymczasowego obszaru i różnicy długości jego boków. Zastosowanie tego współczynnika ma na celu tworzenie obszarów o równomiernym kształcie (zbliżonym do kwadratu).
3. *ruletki* - pozycja wstawienia prostokąta jest losowana zgodnie z zasadą ruletki. Prawdopodobieństwo wylosowania jest proporcjonalne do wielkości tymczasowego obszaru.
4. *XYZ* - pozycja jest losowana zgodnie z rozkładem jednostajnym.

Algorytm symulowanego wyżarzania

Przykładowe rozwiązanie dopuszczalne można obliczyć za pomocą algorytmu pakowania na podstawie losowo wyznaczonego ciągu α. Takie rozwiązanie możne jednak znacznie odbiegać od optymalnego. Pojawia się pomysł, aby generować otoczenie poprzez modyfikację rozpatrywanego rozwiązania i w przypadku gdy bieżące rozwiązanie jest gorsze niż to z sąsiedztwa, zastąpić je nim.
Algorytm symulowanego wyżarzania rozszerza ten pomysł i dodatkowo przyjmuje następujące parametry:

- temperatura początkowa
- funkcja prawdopodobieństwa z jakim są akceptowane rozwiązania gorsze od bieżącego
- schemat schładzania temperatury (zmniejszania wartości)

Wybór tych parametrów wpływa na częstość akceptowania rozwiązań gorszych od bieżącego, a więc na zdolność algorytmu do opuszczania minimów lokalnych, jak i stabilność poszukiwań.

Algorytm symulowanego wyżarzania

Przez $\mathcal{N}(\alpha)$ oznaczamy otoczenie, przez $G(\alpha)$ pole obszaru wewnątrz którego są umieszczone prostokąty, a przez t parametr kontrolny (temperaturę).

$\alpha_{best} \leftarrow \alpha$
repeat
  repeat
    $\beta \leftarrow$ losowy element z otoczenia $\mathcal{N}(\alpha)$
    if $G(\beta) < G(\alpha)$ **then** $\alpha \leftarrow \beta$
    else
      if $e^{\frac{G(\alpha)-G(\beta)}{t}} > random$ **then** $\alpha \leftarrow \beta$
    if $G(\beta) < G(\alpha_{best})$ **then** $\alpha_{best} \leftarrow \beta$
  until zmienić parametr kontrolny
  zmień parametr kontrolny t
until warunek końca

Eksperymenty obliczeniowe

Własności omawianych algorytmów zostały przebadane na podstawie specjalnie przygotowanego programu do pakowania prostokątów.

Specyfikacja

- Implementacja w języku *F#* bez wielowątkowości github.com/michlord/Praktyka-optymalizacji/tree/master/BinPacking
- Program został uruchomiony na komputerze z procesorem AMD 4.1GHz i 8GB RAM
- Parametry alg. symulowanego wyżarzania
  - temperatura początkowa $t_0 = 125$
  - współczynnik zmiany temperatury $\lambda = 0,98$
  - zmiana temperatury co 100 iteracji
  - powrót do temperatury początkowej co 500 iteracji
  - maksymalna liczba iteracji - 5000

Dane wejściowe

Wykonano serię testów dla następujących parametrów

- Ilość prostokątów - 50; Ilość maszyn - 160
- Ilość typów prostokątów - [3, 5, 8, 10, 12, 15, 20]
- Strategie pakowania - [min. obszaru, pole-wymiary, losowa]

Wyniki umieszczono w tabeli. Przyjęto następujące oznaczenia

- Typy - liczba typów prostokątów
- Dokładność - suma pól wszystkich prostokątów podzielona przez pole otrzymanego obszaru
- Pole obszaru - pole obszaru zawierającego upakowane prostokąty
- Krótszy, dłuższy bok - długość krótszego i dłuższego boku pola obszaru
- Ratio - stosunek dłuższego boku do krótszego
- Czas - czas obliczeń w sekundach

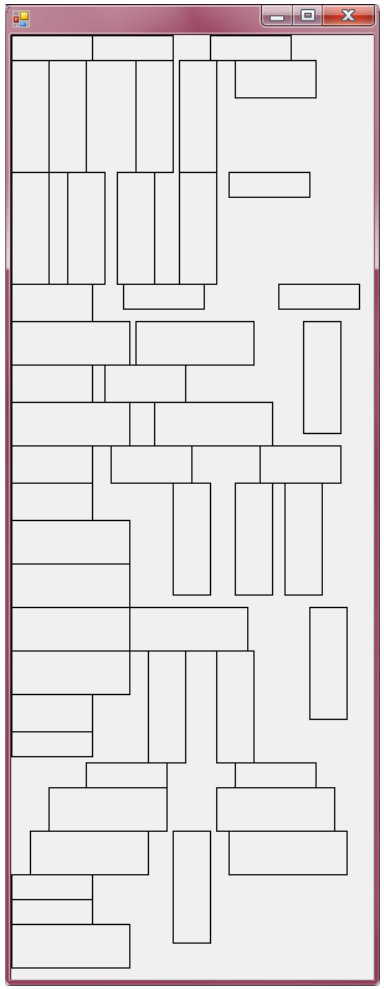
Wyniki obliczeń

Tabela: Strategia minimalnego obszaru

Typy	Dokładność	Pole obszaru	Krótszy bok	Dłuższy bok	Ratio	Czas
3	0.72582547	10176	64	159	2.484375	32.40533
5	0.6454418	7130	46	155	3.369565	32.93213
8	0.63224276	8008	52	154	2.961538	32.24017
10	0.67144928	6900	46	150	3.26087	34.18809
12	0.66198157	4340	28	155	5.535714	34.17906
15	0.65539995	7426	47	158	3.361702	33.93743
20	0.63566308	5580	36	155	4.305556	34.0191

Przykładowe upakowanie

Rysunek: Strategia minimalnego obszaru



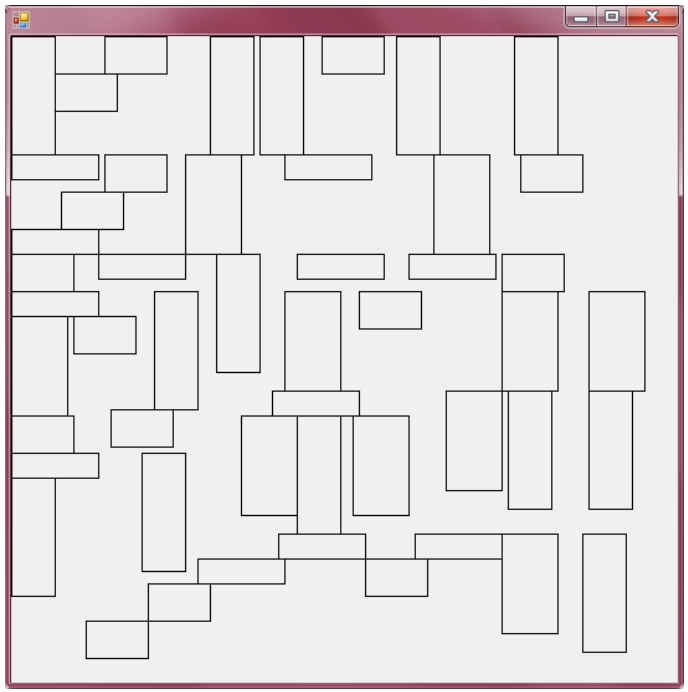
Wyniki obliczeń

Tabela: Strategia współczynnika pole-wymiary

Typy	Dokładność	Pole obszaru	Krótszy bok	Dłuższy bok	Ratio	Czas
3	0.7006058	9409	97	97	1	33.80433
5	0.55073462	4356	66	66	1	32.71974
8	0.64316966	14399	119	121	1.016807	32.67322
10	0.60982973	8281	91	91	1	32.84179
12	0.64555766	8464	92	92	1	32.82862
15	0.58703512	7744	88	88	1	33.57598
20	0.59922251	5402	73	74	1.013699	33.12795

Przykładowe upakowanie

Rysunek: Strategia współczynnika pole-wymiary



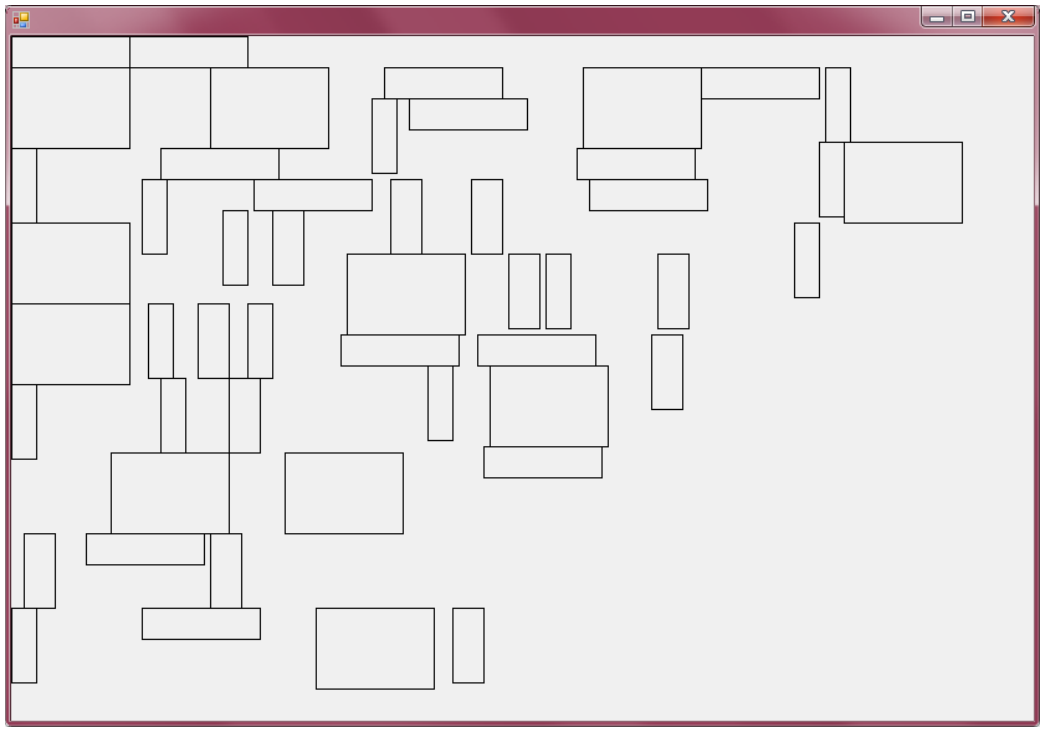
Wyniki obliczeń

Tabela: Strategia losowa

Typy	Dokładność	Pole obszaru	Krótszy bok	Dłuższy bok	Ratio	Czas
3	0.41735141	23975	134	207	1.544776	28.1288
5	0.25686275	17850	89	188	2.11236	29.11801
8	0.26661892	23738	79	265	3.35443	28.92327
10	0.26665419	16032	107	190	1.775701	28.91002
12	0.25339877	20375	140	145	1.035714	29.44653
15	0.23601826	27375	127	174	1.370079	28.42857
20	0.18061538	19500	100	169	1.69	28.92149

Przykładowe upakowanie

Rysunek: Strategia losowa



Wnioski

- Najlepsze wyniki uzyskano stosując strategię minimalnego obszaru. Wartości współczynnika *Ratio* były zdecydowanie większe od wartości otrzymanych dla innych strategii.
- Wyniki strategii współczynnika-wymiary były niewiele gorsze. Natomiast dla tej strategii otrzymano najmniejsze wartości współczynnika *Ratio* bliskie 1.00. Oznacza to, że kształt wyznaczanego obszaru był zbliżony do kwadratu.
- Wartości otrzymane za pomocą strategii losowej były znacząco gorsze od pozostałych. Jedyną zaletą strategii był krótszy czas wykonywania obliczeń.
- Można wywnioskować, że w przypadku problemu harmonogramowania zadań wieloprocesorowych, wybór strategi zależy od relacji kosztu czasu wykorzystania procesorów, a liczbą użytych procesorów.

Literatura

(Bożejko et al., 2016)

Bożejko, W., Kacprzak, Ł., Nadybski, P. and Wodecki, M. (2016). Multi-machine Scheduling with Setup Times.

Computer Information Systems and Industrial Management, 300 – 311.

Dziękujemy za uwagę!