ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1993

Seria: INFORMATYKA z. 24 Nr kol. 1222

Jacek BŁAŻEWICZ
Maciej DROZDOWSKI

SZEREGOWANIE ZADAŃ WIELOPROCESOROWYCH W SYSTEMACH RÓWNOLEGŁYCH

Streszczenie. W pracy omówiono najważniejsze uzyskane wyniki dla problemu szeregowania zadań wieloprocesorowych, tj. zadań żądających do swego wykonania więcej niż jednego procesora w danej chwili czasu. Taki model jest oczywisty dla wielu aplikacji wykorzystujących swój naturalny paralelizm. Okazuje się jednak, że jest to model odmienny od przyjętego powszechnie w teorii szeregowania zadań.

MULTIPROCESSOR TASK SCHEDULING IN PARALLEL SYSTEMS

Summary. In this work we present the most important results for the problems of scheduling multiprocessor tasks, i.e. such ones that each of them requires more than one processor simultaneously. This model seems to be natural for many applications exploiting their naturallz inherited parallelism, This point of view. however, is different than standard in the scheduling theory.

EINORDNUNG VON MULTIPROCESSORAUFGABEN IN PARALLELEN SYSTEMEN

Zusammenfassung. Im Beitrag wurden die wichtigsten Ergebnisse besprochen, die fur das Einordnungproblem der Multipocessoraufgaben (d.i. die Aufgaben die mehr als ein Processor in dem selben Moment zur Ausfuhrung brauchen) erreicht wurden, Dieses Model ist neu in der Theorie der Augabeneinordnung.

Badania zostały częściowo sfinansowane z grantu KBN

1. Wstęp

Wiele istotnych problemów matematycznych można rozwiązać na komputerach pracujących równolegle. Oczywistym celem takiego postępowanie jest skrócenie czasu, w którym uzyskane zostanie rozwiązanie. W teorii szeregowania zadań powszechne jest jednak założenie, że zadania wykonywane są tylko przez jeden procesor w danej chwili czasu [1, 17, 12]. To założenie, w kontekście wielu aplikacji, nie zawsze jest uzasadnione.

Łatwo podać zastosowania z dziedziny matematyki, fizyki, grafiki komputerowej, w których problem można podzielić na niezależne, równolegle rozwiązywane części (np. przy obliczeniach na macierzach) [24, 35]. Przy takich obliczeniach "niezależne" części muszą się jednak ze sobą komunikować. Powoduje to, że z punktu widzenia szybkości wykonania najlepszym rozwiązaniem jest, aby komunikujące się moduły działały jednocześnie. W przeciwnym razie cały program będzie wykonywać się z prędkością, z jaką przydzielany będzie procesor do poszczególnych komunikujących się modułów. Jest to istotne zarówno w systemach ze współdzielną pamięcią [24], jak i w sieciowych systemach komputerowych udostępniających swoją moc obliczeniową poprzez środowiska programistyczne, takie jak Express lub PVM [37, 33].

Inne przykłady aplikacji, w których zadania wymagają więcej niż jednego procesora, to np. wzajemne testowanie się procesorów [20, 26, 29], systemy o podwyższonej niezawodności wykorzystujące "głosowanie" między równolegle działającymi kopiami programu, nad wypracowywanym wynikiem [24].

Problem deterministycznego szeregowania zadań żądających więcej niż jednego procesora w danej chwili czasu można przedstawić za pomocą wielu modeli. Np. zadanie można reprezentować jako zbiór procedur, między którymi występują ograniczenia kolejnościowe wynikające z konieczności komunikacji. W tym modelu przydział komunikujących się procedur do różnych procesorów powoduje opóźnienia komunikacyjne [16, 19, 38]. Model ten nie będzie tu jednak dalej rozważany. Będziemy zakładać, że opóźnienia komunikacyjne zostały uwzględnione w znanym czasie wykonania zadania, Zadania takie nazwijmy wieloprocesorowymi.

W pracy przedstawione zostanie zestawienie najważniejszych wyników uzyskanych dla problemu szeregowania zadań wieloprocesorowych.

Dalsza organizacja pracy jest następująca: Rozdział 2 definiuje problem szeregowania zadań wieloprocesorowych. Rozdział 3, to zestawienie ważniejszych, znanych wyników. Rozdział 4, to zakończenie.

2. Sformułowanie problemu szeregowania zadań wieloprocesorowych

W dalszym ciągu pracy będziemy zakładać, że dany jest zbiór procesorów $\mathcal{P}=\{P_1,...,P_m\}$ składający się z m procesorów połączonych przez współdzielną pamięć lub sieć połączeń komunikacyjnych. W przypadku połączeń komunikacyjnych zakładamy, że moc obliczeniowa węzła nie zależy od faktu przesyłania komunikatów. Zakładamy, że dany jest ybiór dodatkowych zasobów $R=\{R_1,...,R_l\}$, składający się z l typów zasobów, z których każdy dostępny jest w liczbie m_i jednostek, $i=1,...,l$.

Przez $\mathcal{T}$ oznaczać będziemy zbiór n zadań. Ponieważ rozpowszechnionych jest kilka modeli dla jednoczesnego żądania procesorów przez zadania, dlatego najpierw opiszemy te modele [38]. W przypadku, gdy istotna jest liczba żądanych procesorów, wyróżnia się modele:

1. "$size_j$" – każde zadanie j żąda do swego wykonania określonej liczby $size_j$ procesorów jednocześnie. Oznaczmy takie zadanie T_j^k, gdy $k=size_j$, a jego czas wykonania oznaczmy przez t_j^k. Zbiór wszystkich zadań żądających dokładnie k procesorów jednocześnie oznaczmy przez T^k.
2. "$cube_j$" – każde zadanie żąda określonej liczby procesorów, która jest potęgą dwójki. Ten model jest szczególnym przypadkiem modelu "$size_j$" i dotyczy szeregowania na hiperkostkach (ang. hypercube) procesorów.
3. "any" – każde zadanie może być wykonywane przez dowolny podzbiór procesorów, ale czas wykonania zależy od liczby przydzielonych procesorów. Liczba jednocześnie wykorzystanych procesorów nie może być zmieniona w czasie wykonywania zadania. Model ten jest uogólnieniem modelu "$size_j$". Zależność między czasem wykonania i liczbą przydzielonych procesorów jest, w ogólności, dowolna. Dla pewnych prostych przypadków równoległości [40] zakłada się, że jest to zależność odwrotnie proporcjonalna. W przypadku, gdy istotny jest *zbiór* żądanych procesorów (a nie liczba), wyróżniamy dwa modele:
4. "fix_j" – każde zadanie j żąda do swego wykonania ustalonego zbioru fix_j procesorów jednocześnie. Oznaczmy takie zadanie T_j^D, gdzie $D=fix_j$, a jego czas wykonania – t_j^D. Zbiór wszystkich zadań żądających dokładnie zbioru D procesorów oznaczmy T^D.
5. "set_j" – każde zadanie j może być wykonywane przez wiele alternatywnych zbiorów procesorów. Zbiór set_j alternatywnych zbiorów jest znany, a procesory z każdego takiego zbioru D_i są żądane jednocześnie. Model ten jest uogólnieniem modelu "fix_j".

Z każdym zadaniem j związany jest zbiór dodatkowych parametrów. Są one takie same, jak w klasycznej deterministycznej teorii szeregowania zadań. Są to: r_j – moment pojawienia się w systemie, d_j – linia krytyczna, $R(T_j)=[R_1(T_j),...,R_l(T_j)]$ – wektor żądań zasobowych, gdzie ~ oznacza, że sposób żądania procesorów nie jest w danym kontekście istotny. Tak samo też definiuje się *ograniczenia kolejnościowe* między zadaniami oraz *podzielność* lub *niepodzielność* zbioru zadań.

Kryteria optymalności uszeregowania są takie same, jak w klasycznej teorii szeregowania zadań. Są to: C_{max} – długość uszeregowania, L_{max} – maksymalne spóźnienie zadań, $\sum C_j$ – średni czas przepływu i $\sum w_j C_j$ – średni ważony czas przepływu.

Pojęcie *uszeregowania* jest rozumiane tak samo, jak w klasycznej teorii szeregowania zadań, ale wymaga się dodatkowo, by zadania ze zbioru T^k otrzymały do swego wykonania k procesorów jednocześnie, a zadania ze zbioru T^D – wszystkie procesory ze zbioru D.

Do zaspisu rozważanych problemów szeregowania zadań wieloprocesorowych wykorzystana zostanie standardowa trójpolowa notacja podana w [25] wraz z rozszerzeniami określonymi w [13] i [38]. Wieloprocesorowość zadań uwzględniona zostaje w tym zapisie przez dodanie w drugim polu słowa kluczowego, odpowiadającego wymienionym wcześniej modelom żądania procesorów. Np. $P3|fix_j,pmtn|C_{max}$ – oznacza problem szeregowania na trzech procesorach zadań podzielonych, żądających określonego zbioru procesorów, gdzie długość uszeregowania jest kryterium optymalności.

3. Przegląd wyników

3.1. Modele: $size_j$, $cube_j$, any

W przypadku szeregowania zadań niepodzielnych problem jest NP-trudny już dla klasycznego sformułowania i dla dwóch procesorów [28]. Dlatego dalsze poszukiwania skupiły się na wyznaczeniu wyraźnej granicy między problememi łatwymi i trudnymi. W przypadku tych ostatnich starano się podać heurystyki o określonych gwarancjach jakości. W przypadku uszeregowań podzielonych problem jest również NP-trudny w ogólności [22]. Wydaje się jednak, że dla znacznie szerszej klasy istotnych przypadków udało się podać algorytmy wielomianowe. Zestawienie wyników zostało zawarte w tabeli 1.

Tabela 1

Szeregowanie zgodnie z modelami $size_j$, $cube_j$, any

Problem	Rezultat	Referencja
	Uszeregowanie niepodzielne	
$P3\|size_j \in \{1,2\},prec\|C_{max}$	NPh	[32]
$P\|size_j \in \{1,...,k\},prec\|C_{max}$	$S_{LS}=(2m-k)/(m-k+1)$	["]
$P2\|size_j,prec\|C_{max}$	$O(n^2)$ (Coffman-Graham)	["]
$P\|size_j \in \{1,k\},p_j=1\|C_{max}$	O(n)	[6]
$P\|size_j \in \{1,...,k\},p_j=1\|C_{max}$	0(n)	["]
$P2\|size_j,chain\|C_{max}$	sNPh	[23]
$P2\|any\|C_{max}$, $P3\|any\|C_{max}$	NPh, pseudowielomianowy	["]
$P4\|any\|C_{max}$	?	
$P5\|size_j\|C_{max}$	sNPh	[23]
$P\|any, prec\|C_{max}$	$S_{LS}=k+m(m-k)/m$	[40]
	$S_{ECT}<\ln(k)+1$	["]
	$S_{ECT}\leq 3-2/m$	[41]
	Uszeregowanie podzielne	
$P\|size_j \in \{1,k\},pmtn\|C_{max}$	0(n)	[6]
$P\|size_j \in \{1,...,k\},pmtn\|C_{max}$	programow. liniowe	["]
$P\|cube_j,pmtn\|C_{max}$	$0(n^2(\log n+\log(\max t_j)))$	[15]
	$0(n\log n(\log n+\log(\max t_j)))$	[27]
	$0(n^2m^2)$	[36]
$Q\|size_j \in \{1,k\},pmtn\|C_{max}$	0(nm+nlogn)	[7]
$Q\|cube_j,pmtn\|C_{max}$	0(nm+nlogn)	[8]
$Qm\|cube_j \in \{1,...,k\},pmtn\|C_{max}$	programow. liniowe	["]
$P\|size_j,pmtn\|C_{max}$	NPh	[22]
$P\|any,pmtn\|C_{max}$	sNPh	[23]
$Pm\|any,pmtn\|C_{max}$	NPh, pseudowielomianowy	["]
$P\|size_j \in \{1,2\},pmtn,res1^{-}1\|C_{max}$	0(nm)	[11]
$P\|cube_j,pmtn\|L_{max}$	programow. liniowe	[34]
$Pm\|size_j,pmtn\|L_{max}$	programow. liniowe	[9]

Oznaczenia: NPh – NP-trudny, sNPh – silnie NP-trudny, O(f) – funkcja złożoności algorytmu jest rzędu f, S_A – bezwzględne oszacowanie jakości algorytmu heurystycznego A.

3.2. Modele: fix_j, set_j

Problemy szeregowania zadań wieloprocesorowych opisanych modelami "fix_j" i "set_j" są blisko związane z problemami kolorowania grafów, a więc problemami, które należą do klasycznie trudnych zagadnień optymalizacji kombinatorycznej. W tym kontekście nietrudno przewidzieć, że większość rozważanych problemów, to problemy NP-trudne i silnie NP-trudne, a reprezentacja problemów za pomocą grafów jest szczególnie częsta. Najczęściej zbiór zadań reprezentuje się za pomocą grafu niezgodności (*ang.* incompatibility graph), w którym zadania odpowiadają węzłom, a krawędzie łączą węzły (zadania), które nie mogą być wykonywane jednocześnie. Alternatywnie można także posłużyć się tzw. grafem transferu plików (*ang.* file transfer graph), w którym węzły odpowiadają procesorom, a krawędzie odpowiadające zadaniom łączą jednocześnie wykorzystywane procesory i mają wagę równą czasowi wykonania zadania. Znane wyniki zostały zestawione w tabeli 2.

Tabela 2

Szeregowanie zgodnie z modelami set_j, fix_j

Problem	Rezultat	Referencja
	Uszeregowanie niepodzielne	
$P\|fix_j\|C_{max}$	algorytm B&B	[14]
$P\|fix_j\|C_{max}$ $\|fix_j\|=2$	$S_{LF}=4(d-1)/d$	[29]
	$S_{LF}\leq 3$ $(d\leq 5)$	["]
	$S_{LF}\leq 2$ (FTG typu binomial)	["]
$P\|fix_j,p_j=1\|C_{max}$ $\|fix_j\|=2$	NPh w ogólności, ale wielomianowy, gdy FTG jest dwódzielny, jednocykliczny gwiazdą, drzewem, gąsienicą, cyklem lub ścieżką	[18]
$P\|fix_j\|C_{max}$ $\|fix_j=2$	wielomianowy, gdy FTG jest cyklem lub ścieżką	[18]
$P\|fix_j\|C_{max}$ $\|fix_j=2$	$C_{max}^{LS}\leq 2C_{max}^{*}+\max\{0,\max\{t_j\}\}(1-2/d)$	[18]
$P\|fix_j\|C_{max}$ $\|fix_j\|\epsilon\{1,2\}$	NPh, jeśli FTG jest gąsienicą z chociaż jedną pętlą lub gwiazdą z pętlami w każdym niecentralnym wierzchołku	[30]
$P\|fix_j\|C_{max}$	$O(dn^2)$, gdy IG jest grafem typu (*ang.*) comparability	[5]
$P\|fix_j\|C_{max}$	algorytm B&B	["]

$P3\|fix_j\|C_{max}$	sNPh	[10]
	$s_{NS}<4/3$	["]
$P3\|fix_j\|C_{max}$	zadania danego typu wykonywane są blokowo, tj. bez wtrąceń zadań innych typów NPh, pseudowielomianowy	[39]
$Pm\|fix_j,p_j=1\|C_{max}$	O(n)	["]
$P\|fix_j,p_j=1\|C_{max}$	sNPh (3–kolorowanie)	["]
$P2\|chain,fix_j,p_j=1\|C_{max}$	sNPh	["]
$P2\|fix_j,r_j\|C_{max}$	sNPh	["]
$P2\|fix_j,p_j=1\|\sum w_jC_j$	O(nlogn)	[21]
$P2\|fix_j\|\sum C_j$	NPh (podział zbioru)	[39]
$P2\|fix_j\|\sum w_jC_j$	sNPh (3–podział)	["]
$P3\|fix_j\|\sum C_j$	sNPh (P3 fix_j C_{max})	["]
$P\|fix_j,pj=1\|\sum C_j$	sNPh (P $fix_j,p_j=1$ C_{max})	["]
$P2\|chain,fix_j,p_j=1\|\sum C_j$	sNPh (P2 chain,$fix_j,p_j=1$ C_{max})	["]
$P2\|set_j\|C_{max}$	pseudowielomianowy	[3]
P3 set_j $Cmax$ $T^{1,3}=\emptyset$	pseudowielomianowy	["]
$P\|set_j\|C_{max}$	$S_{SPT}=m$	["]
	Uszeregowanie podzielone	
$P\|fix_j,pmtn\|C_{max}$	NPh (multikolorowanie)	[31]
$P2\|fix_j,pmtn\|C_{max}$	O(n)	[2]
$P3\|fix_j,pmtn\|C_{max}$	O(n)	["]
$P4\|fix_j,pmtn\|C_{max}$	O(n)	["]
$P3\|fix_j,pmtn,resl.1\|C_{max}$	O(n)	["]
$Pm\|set_j,pmtn\|C_{max}$	programowanie liniowe	[3]
$Pm\|set_j,pmtn\|L_{max}$	programowanie liniowe	[4]
$Pm\|set_j,pmtn,r_j\|L_{max}$	programowanie liniowe	["]

Oznaczenia: NPh – NP-trudny, sNPh – silnie NP-trudny, O(f) – funkcja złożoności algorytmu jest rzędu f, S_A – bezwzględne oszacowanie jakości algorytmu heurystycznego A, B&B – algorytm podziału i ograniczeń, d – maksymalny rząd dowolnego wierzchołka w grafie, IG – graf niezgodności, FTG – graf transferu plików.

4. Zakończenie

W pracy przedstawiono bieżący stan dziedziny szeregowania zadań, jaką jest szeregowanie zadań wieloprocesorowych. Analiza złożoności obliczeniowej pokazuje, że chociaż większość problemów, to problemy obliczeniowo trudne, to dla pewnych istotnych przypadków szczególnych można podać wielomianowe algorytmy dokładne. Można wskazać wiele kierunków dalszych poszukiwań dla teorii szeregowania zadań w środowisku równoległych systemów komputerowych. Mogą one dotyczyć uwzględnienia innych kryteriów optymalności, dodatkowych zasobów lub szeregowania z opóźnieniami komunikacyjnymi.

LITERATURA

[1] Baker K.R., Introduction to sequencing and scheduling, John Willey & Sons, 1974

[2] Bianco L., Błażewicz J., Dell'Olmo P., Drozdowski M., Scheduling preemptive multiprocessor tasks on dedicated processors, zaaokceptowane w Performance Evaluation.

[3] Bianco L., Błażewicz J., Dell'Olmo P.,Drozdowski M., Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors, raport Instytutu Informatyki Politechniki Poznańskiej R-92/045, 1992.

[4] Bianco L., Błażewicz J., Dell'Olmo P., Drozdowski M., Preemptive scheduling of multiprocessor tasks on the dedicted processor system subject to minimal latenss, IPL 46, 1993, 109-113.

[5] Bianco L., Dell'Olmo P., Speranza M.G., Nonpreemptive scheduling of independent tasks with dedicated resources, Tech. Rept. R.320 of IASI CNR, Rome, June 1991.

[6] Błażewicz J., Drabowski M., Węglarz J., Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans. Comput., C-35, 1986, 389-393.

[7] Błażewicz J., Drozdowski M., Schmidt M., de Werra D., Scheduling independent two processor tasks on a uniform duo-processor systems, Discrete Applied Mathematics 28, 1990, 11-20.

[8] Błażewicz J., Drozdowski M., Schmidt G., de Werra D., Scheduling independent multiprocessor tasks on a uniform k-processor systems, raport Insytutu Informatyki Politechniki Poznańskiej R-92/030, 1992.

[9] Błażewicz J., Drozdowski M., WEglarz J., de Werra D., Scheduling independent multiprocessor tasks before deadlines, raport Instytutu Informatyki Politechniki Poznańskiej R-92/043, 1992.

[10] Błażewicz J., Dell'Olmo P., Drozdowski M., Speranza M.G., Scheduling multiprocessor tasks on three dedicated processors, IPL 41, 1992, 275-280.

[11] Błażewicz J., Ecker K., Scheduling multiprocessor tasks under unit resource constraints, Int. Conf. on Optimization Techniques and Applications, Singapore, April 1987, 161-169.

[12] Błażewicz J., Ecker K., Schmidt G., Węglarz J., Scheduling in Computer and Manufacturing Systems, Springer-Verlag, 1993.

[13] Błażewicz J., Lenstra J.K., Rinnoy Kan A.H.G., Scheduling subject to resource constraints: classification and complexity, Discrete Applied Mathematics 5, 1983, 11-24.

[14] Bozoki G., Richard J.P., A branch-and-bound algorithm for the continuous-process task shop scheduling problem, AIIE Trans. 2, 1970, 246-252.

[15] Chen G.I., Lai T.H., Preemptive scheduling of independent jobs on hypercube, IPL 28, 1988, 201-206.

[16] Chretienne P., Task scheduling with interprocessor communication delays, EJOR 57 (1992), 348-354.

[17] Coffman Jr., E.G.(edytor), Teoria szeregowania zadań, WNT, Warszwa 1980.

[18] Coffman Jr. E.G. Grey M.R., Johnson D.S. and LaPaugh A.S., Scheduling file transfers. SIAM J.Comput, wol. 14, No.3, 1985, 744-780.

[19] Colin J.Y., Chretienne P., C.P.M. scheduling with small communication delays and task duplication, Operations Research, vol. 39, July-August 1991, No. 4, 680-684, 1991.

[20] DalCin M., Dilger E., On the diagnosibility of self-testing multiprocessor systems, Microprocessing Microprogramming, vol.7, No. 3, 1981, pp.177-184.

[21] Dobson G., Karmarkar U.S., Simultaneous resource scheduling to minimize weighted flow times, Operations Research vol. 37, No 4, July-August 1989, 592-600.

[22] Drozdowski M., Problemy i algorytmy szeregowania zadań wieloprocesorowych, praca doktorska, Instytut Informatyki Politechniki Poznańskiej, 1992.

[23] Du J., Leung J.Y-T., Complexity of scheduling parallel task systems, SIAM J.Discrete math. 2, 1989, 473-487.

[24] Gehringer E., Siewiorek D., Segall Z., Parallel processing. The Cm* experience, Digital Press, 1987.

[25] Graham R.I., Lawler E.L., Lenstra J.K., RinnoyKan A.H.G., Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5, 1979, 287-326.

[26] Hakimi S.L., Amin A.T., Characterization of Connection Assignment of Diagnosable Systems, IEEE Trans. on Comput., vol. C-23, 1974, 86-88.

[27] Hoesel van C.P.M., Preemptive scheduling on a hypercube, Report 8963/A, Econometrics Institute, Erasmus University Rotterdam.

[28] Karp R.M., Reducibility among combinatorial problems, in R.E. Miller and J.W. Tatcher (eds.) Complexity of Computer Computation, Plenum Press, New York 1972.

[29] Krawczyk K., Kabale M., An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Trans. Comput. C-34, 1985, 869-872.

[30] Kubale M., The complexity of scheduling independent two-processor tasks on dedicated processors, IPL 24, 1987, 141-147.

[31] Kubale M., Podzielne uszeregowanie zadań dwuprocesorowych na procesorach dedykowanych, Zeszyty Naukowe Politechniki Śląskiej, Seria: Automatyka z.100, nr 1082, 1990, 145-153.

[32] Lloyd E.L., Cocurrent task systems, Operations Reserch vol 29, No. 1, Jan-Feb 1981, 189-201.

[33] Mościński J., Boryczko K., Kitowski J., Pogoda M., Alda W., Bubak M., Dzwinel W., Słota R., Noga M., Distributed molecular dynamics on a cluster of workstations and CONVEX supercomputer, Proceedings of the Convex User Group Worldwide Conference, Richardson, Texas, March 21-25, 1993.

[34] Plehn J., Preemptive scheduling of independent jobs with release times and deadlines on a hypercube, IPL 34, No.3, 1990, 161-166.

[35] Seitz C., The Cosmic Cube, Comm. of the ACM, vol. 28, No 1, 1985.

[36] Shen X., Reingold E.M., Scheduling on a hypercube, Information Processing Letters, vol. 40, No. 6, (1991), 323-328.

[37] Sunderam V.S., Emory U., PVM: A framework for parallel distrubuted computing, Concurrency: Practice & Experience, vol. 2, No. 4, Dec. 1990, 315-339.

[38] Veltman B., Lageweg B.J., Lenstra J.K., Multiprocessor scheduling with communication delays, Parallel Computing 16, 1990, 173-182.

[39] Hoogeveen J.A., Velde S.L.van de, Veltman B., ParTool: A parallel Processing Development Environment, Complexity of Scheduling Multiprocessor Task with prespecified Processor Allocations. CWI Centre for Mathematics and Computer Science, BS-R9211, Amsterdam, The Netherlands.

[40] Wang Q., Cheng K.H., List scheduling of parallel tasks, IPL 37 (1991), 291-297.

[41] Wang Q., Cheng K.H., A heuristic of scheduling parallel tasks and its analysis. SIAM J. on Comput., 1992, vol 21, No. 2, 281-294.

Recenzent: Dr hab. inż. Stanisław Kozielski Profesor Pol. Śląskiej

Wpłynęło do redakcji 20 września 1993 r.

Abstract

Many important mathematical problems can be solved by machines working in parallel. An obvious goal of such an approach is the reduction of the time in which a final solution can be found. Over the past decades it was commonly assumed in the scheduling theory that a task can be executed bu only one processor at a time. This assumption in the context of many applications is not so obvious and requires a new look at as well as new algorithms are required.

In this work we present the summary of the state-of-the-art across the problems of scheduling multiprocessor tasks, i.e. such ones that each of them requires more than one processor at the time. Computational complexity analysis shows that majority of the problems are computationally hard. However, some important cases can be solved exactly by simple polynomial-time algorithms or by linear programming. Results are collected in Table 1 and Table 2.