Linear time approximation schemes for parallel processor scheduling

Y. Kopidakis,
D. Fayard,
and
V. Zissimopoulos

Université de Paris Sud
L.R.I, CNRS-URA 410
Centre d'Orsay, 91405 Orsay, France
e_mail:kopidaki@lri.lri.fr

Abstract

We present a general framework for approximation schemes on parallel processor scheduling. We propose ϵ-approximation algorithms for scheduling on identical, uniform and unrelated machines when the number of processors is fixed. For each of the three problems considered, we perform grouping on job processing times in order to produce a transformed scheduling instance where the number of distinct task types is bounded. We optimally solve the corresponding mixed integer program and we prove that the optimal makespans for the initial and the transformed problems can differ at most by a factor of $1+\epsilon$. The complexity of all ϵ-approximation algorithms is $O(n)$, where n is the number of jobs to be scheduled.

1 Introduction

The problem of scheduling parallel processors in order to minimize schedule completion time has been extensively studied in various formulations. We are interested in the case where n independent tasks are to be scheduled on a fixed number of processors m. We consider the following classical NP-complete scheduling problems: identical processors where task execution time is the same on any processor (known as the multiprocessor scheduling problem), uniform processors where task execution cost depends on the computational speed factor of each processor and unrelated machines where the execution time of a task depends on the processor in the general way.

In the study of an NP-complete problem, the strongest possible type of result is an approximation scheme. An ϵ-approximation algorithm resolves the problem within an arbitrarily small precision ϵ, fixed in advance as desired. Formally, let $\mathcal{I}$ be the set of instances of a NP-complete minimization problem. Let $I \in \mathcal{I}$ and $OPT(I)$ the value of the optimal solution for I. We say that an algorithm A is a polynomial time approximation scheme for the problem, if given any $\epsilon > 0$: $A(I) \leq (1+\epsilon)\, OPT(I), \ \ \forall I \in \mathcal{I}$, where $A(I)$ is the value of the solution for I returned by A in $O(p(n))$ time and $p(n)$ a polynomial depending on n. In addition, if algorithm A runs in $O(p(n, \frac{1}{\epsilon}))$ time, where $p(n, \frac{1}{\epsilon})$ is a polynomial on both n and $\frac{1}{\epsilon}$, we say that algorithm A is a fully polynomial time approximation scheme.

In this paper, we propose a general approximation framework which we apply for each of the scheduling problems considered when the number of processors is fixed. The time complexity of the general approximation scheme proposed is linear on the number of jobs to be scheduled. Although approximation schemes exist for identical and uniform processor scheduling in the general case, that is when we do not fix the number of processors, the interest of our approach lies in its asymptotically reduced time complexity. Furthermore, the extension of the general framework to the problem of unrelated machines, provides a significant complexity improvement compared to existing approximation schemes for the case of fixed number of processors.

2 Multiprocessor scheduling

The problem of finding a minimum completion time schedule of n independent jobs on m identical processors is one of the most well-studied NP-complete problems. In [9], Sahni presented a fully polynomial time approximation scheme for fixed number of processors, running in $O(n(\frac{n^2}{\epsilon})^{m-1})$ time. The notion of dual approximation algorithms has been introduced by Hochbaum and Schmoys [2] and has provided the first approximation scheme for the general problem, having a time complexity of $(O((\frac{n}{\epsilon})^{\frac{1}{\epsilon^2}})$. This complexity was later improved by Leung [8] to $(O((\frac{n}{\epsilon})^{\frac{1}{\epsilon}log\frac{1}{\epsilon}})$.

In what follows, we design a polynomial time approximation scheme for multiprocessor scheduling on a fixed number of processors which runs in $O(n\, log\frac{-log h_{min}}{log(1+\epsilon)})$ time.

Initially, we consider a mixed integer programming formulation for the problem. Let $t_i,\ i = 1, ..., n$ represent the processing time of job i, $x_{ij},\ i = 1, ..., n,\ j = 1, ..., m$, a 0-1 variable indicating whether task i is assigned on processor j and C_{max} an auxiliary variable representing schedule completion time. Then, any instance of the problem can be formulated as (MP):
$min\ C_{max}\ \ s.t:\ \sum_{i=1}^{n} t_i x_{ij} \leq C_{max},\ j = 1, .., m,$
$\sum_{j=1}^{m} x_{ij} = 1,\ i = 1, .., n\ \ and$
$x_{ij} \in \{0, 1\},\ i = 1, .., n,\ j = 1, .., m,\ \ C_{max} \in R$

We transform MP into another multiprocessor scheduling instance where all tasks are distributed into k distinct task types using geometric grouping on task processing time. The description of the corresponding linear programming formulation for the transformed instance MP' will be provided in the lines to follow. Let t_i' the transformed processing time for task i in MP'. Initially, we normalize processing times in the interval $(0, 1]$, dividing t_i, $\forall i = 1, .., n$ by $t_{max} = max\{t_i,\ i = 1, .., n\}$. Next, following the lines of research of [1], [5] and [8], we partition the interval $(0, 1]$ into k subintervals as follows:
$[t_{min} = (1+\epsilon)^{-k}, (1+\epsilon)^{-(k-1)}]$, $((1+\epsilon)^{-(k-1)}, (1+\epsilon)^{-(k-2)}]...((1+\epsilon)^{-q}, (1+\epsilon)^{-(q-1)}]...((1+\epsilon)^{-1}, 1]$.

Clearly: $k = \lceil\frac{-log\ t_{min}}{log(1+\epsilon)}\rceil$ (1)

We define t_i' as follows:
$t_i' = t_q = (1+\epsilon)^{-(q-1)}$, (2)
$if\ (1+\epsilon)^{-q} < \frac{t_i}{t_{max}} \leq (1+\epsilon)^{-(q-1)},\ q = 1, ..., k$

Let b_q denote the number of tasks regrouped in task type q. Obviously, $\sum_{q=1}^{k} b_q = n$. If $x_{qj},\ q = 1, .., k,\ j = 1, .., m$ is now a non negative integer indicating the number of tasks of type q assigned on processor j, the optimal solution is provided by the following mixed integer program, (MP'):
$min\ C_{max}'\ \ s.t:\ \sum_{q=1}^{k} t_q' x_{qj} \leq C_{max}',\ \ j = 1, .., m,$
$\sum_{j=1}^{m} x_{qj} = b_q,\ \ q = 1, .., k\ \ and$
$x_{qj} \in N,\ \ q = 1, .., k,\ j = 1, .., m,\ \ C_{max}' \in R$

Proposition 1 *When the number of processors m is fixed, MP' can be optimally solved in constant time.*

Proof: Let's consider the mixed integer programming formulation for MP'. The maximum number of variables in the formulation is $km + 1$, since there are at most km assignment variables x_{qj} and one auxiliary variable C_{max}. The number of constraints is $m + k$. In the case where the number of processors m is fixed, both the number of variables and constraints are bounded above by constants not depending on n. Consequently, we can resolve MP' in constant time (see the work of Lenstra [6]). However, since the number of variables and constraints depends on k and m, the constant time needed to optimally solve the program, is in the worst case exponential on the number of machines and on the number of distinct task types. □

In addition, we claim that the minimum makespans for the initial and the transformed instances differ at most by a factor of $1 + \epsilon$:

Proposition 2 *If $OPT(MP)$ and $OPT(MP')$ are the optimal solutions for MP and MP' respectively, we have:* $OPT(MP) \leq OPT(MP') \leq (1+\epsilon)\ OPT(MP)$

Proof: In transformation (2), remark that $t_i \leq t_i'$, $\forall i = 1, ..., n$. Thus, the inequality $OPT(MP) \leq OPT(MP')$ is straightforward.

For the second part, in solution $OPT(MP)$, we transform each t_i into t_i'. Let $FEAS(MP')$ the corresponding feasible solution for MP'. By the grouping procedure, when $t_i' = (1+\epsilon)^{-q}$, then $t_i > (1+\epsilon)^{-(q+1)}$. Consequently: $\frac{t_i'}{t_i} < \frac{(1+\epsilon)^{-q}}{(1+\epsilon)^{-(q+1)}}$, $\forall i = 1, ..., n$ and thus:
$t_i' < (1+\epsilon)\, t_i \quad , \ \forall i = 1, ..., n$ (3)

The above relation bounds the size growth of each transformed task length and consequently the size growth of the corresponding solution. In fact, let r be the processor with the heaviest load in the optimal solution of MP. Let $T_r \subseteq \{1, .., n\}$ denote the set of jobs assigned to r and $h_r = \sum_{i \in T_r} t_i$ the total load for r. Clearly, $OPT(MP) = h_r$. When we transform each t_i into t_i', for $FEAS(MP')$ we consider two cases :

a) processor r continues to have the heaviest load.
Then, $FEAS(MP') = h_r'$, where $h_r' = \sum_{i \in T_r} t_i'$. However, from (3), $h_r' < \sum_{i \in T_r} (1+\epsilon)\, t_i = (1+\epsilon)\, h_r = (1+\epsilon)\, OPT(MP)$ and finally:
$FEAS(MP') \leq (1+\epsilon)\, OPT(MP)$.

b) another processor s has now the heaviest load.
Then, $FEAS(MP') = h_s'$. Clearly, in $OPT(MP)$, we have: $h_s \leq h_r$. Exactly as in case a), we have $h_s' < (1+\epsilon)\, h_s \Rightarrow h_s' < (1+\epsilon)\, h_r \Rightarrow h_s' < (1+\epsilon)\, OPT(MP)$ and finally: $FEAS(MP') \leq (1+\epsilon)\, OPT(MP)$.

Thus, for both cases, we have shown the following: $FEAS(MP') < (1+\epsilon)\, OPT(MP)$. Since $OPT(MP')$ is the optimal solution of MP', we have: $OPT(MP') \leq FEAS(MP')$ and finally:
$OPT(MP') \leq FEAS(MP') < (1+\epsilon)\ OPT(MP)$
which proves the second part and the proposition. □

We can now describe the approximation scheme for multiprocessor scheduling with a fixed number of processors.

Initially, we normalize task processing times into interval (0,1], dividing by $t_{max} = max\{t_i,\ i = 1,..,n\}$. This step requires $O(n)$ time. Next, for $k = \lceil \frac{-log\ t_{min}}{log(1+\epsilon)} \rceil$, we get MP' with $t_i' = (1+\epsilon)^{-(q-1)},\ q = 1,...,k$ by grouping task lengths into k distinct task types as indicated by equation (2). The grouping procedure can be performed in $O(n\ log\ k) = O(n\ log\ \frac{-log\ t_{min}}{log(1+\epsilon)})$ time. Then, using proposition 1, we can find the optimal solution of MP' in constant time. Finally, we transform in linear time the determined solution into a feasible one for MP by changing t_i' into initial t_i and we return total height A as solution cost.

Since $t_i \leq t_i'$, A is no greater than its corresponding solution for MP' and using proposition 2, we have: $A \leq OPT(MP') \leq (1+\epsilon)\ OPT(MP)$. Thus, the above algorithm is an approximation scheme for multiprocessor scheduling with fixed number of processors. Its time complexity is $O(n\ log \frac{-log\ t_{min}}{log(1+\epsilon)})$.

3 Uniform processors

In the problem of scheduling uniform processors, we consider m machines that run at different speeds. Let s_j the speed factor for processor j. Then, for the processing time of job i on processor j we have: $t_{ij} = \frac{t_i}{s_j},\ \ i = 1,..,n,\ j = 1,..,m$. Again, the objective function is the minimization of schedule completion time. A polynomial time approximation scheme is provided by Horowitz and Sahni [4] in the case of fixed number of processors, having a complexity of $O((10^{-log\epsilon}n^2)^{m-1})$. Hochbaum and Schmoys [3] presented a dual approximation scheme for the general problem, running in $O(\frac{m\ n^{(\frac{10}{\epsilon^2}+4)}}{\epsilon^2})$ time. We propose a polynomial time approximation scheme for fixed number of processors which runs in $O(n\ log \frac{-log\ t_{min}}{log(1+\epsilon)})$ time.

We formulate the problem as an integer program (UP):
$min\ \ C_{max}\ \ s.t:\ \sum_{i=1}^{n} \frac{t_i}{s_j} x_{ij} \leq C_{max},\ \ \ j = 1,..,m,$
$\sum_{j=1}^{m} x_{ij} = 1,\ \ \ i = 1,..,n\ \ and$
$x_{ij} \in \{0,1\},\ i = 1,..,n,\ j = 1,..,m,\ \ C_{max} \in R$

Obviously, working in exactly the same way as in the previous section we can formulate UP' by grouping all jobs into k constant task types. Then, when the number of processors is fixed, we can optimally solve UP' in constant time. The claim that $OPT(UP) \leq OPT(UP') \leq (1+\epsilon)\ OPT(UP)$ can still be proved. Consequently, the algorithm described in the previous section is an approximation scheme for scheduling on a fixed number of uniform processors.

4 Unrelated machines

In scheduling unrelated machines, processing time depends on both the job and the processor to be used. Thus, given t_{ij} the length of task i on processor j, $i = 1,..,n,\ j = 1,..,m$, we are asked to find the minimum completion time assignment of tasks to processors. Horowitz and Sahni [4] presented the first approximation scheme for the case where the number of processors is fixed. Its time complexity is $O((10^{-log\epsilon}n^2)^{m-1})$. In [7], Schmoys, Lenstra and Tardos provided an approximation scheme for fixed number of processors, requiring the resolution of $(n+1)^{\frac{m}{\epsilon}}$ linear programming relaxations. In the same work, a negative result is proved, excluding the existence of a polynomial ϵ-approximation algorithm for the general problem for any $\epsilon < \frac{1}{2}$, unless $P = NP$. In what follows, we present a polynomial time approximation scheme for unrelated machines scheduling for fixed number of processors which runs in $O(n\ m\ log \frac{-log\ t_{min}}{log(1+\epsilon)})$ time.

Consider the following mixed integer programming formulation for unrelated machines scheduling, (UM):
$min\ \ C_{max}\ \ s.t: \sum_{i=1}^{n} t_{ij} x_{ij} \leq C_{max},\ \ \ j = 1,..,m,$
$\sum_{j=1}^{m} x_{ij} = 1,\ \ \ i = 1,..,n\ \ and$
$x_{ij} \in \{0,1\},\ i = 1,..,n,\ j = 1,..,m,\ \ C_{max} \in R$

We normalize t_{ij} into interval $(0,1]$ dividing by $t_{max} = max\{t_{ij},\ i = 1,..,n,\ j = 1,..,m\}$. Again, we partition the interval $(0,1]$ into k subintervals:
$[t_{min} = (1+\epsilon)^{-k}, (1+\epsilon)^{-(k-1)}]\ ,\ ((1+\epsilon)^{-(k-1)}, (1+\epsilon)^{-(k-2)}]...((1+\epsilon)^{-q}, (1+\epsilon)^{-(q-1)}]...((1+\epsilon)^{-1}, 1]$
where $k = \lceil \frac{-log\ t_{min}}{log(1+\epsilon)} \rceil$.

However, in the case of unrelated machines it does not suffice to consider only k distinct task types. The reason is that two jobs regrouped in the same task type on one processor (their processing times are quite close on this processor), do not necessarily belong to the same task type on other processors (their processing times may differ significantly elsewhere). In fact, we consider separate grouping into k types on each of the m processors, which totals at most k^m distinct task types.

Formally, in UM each job i is characterized by a m-dimensional processing time vector $(t_{i1}, t_{i2}, ..., t_{im})$. Since we perform geometric grouping with parameter k separately for each processor, in the transformed instance UM' (to be described later in detail) job i will be characterized by a m-dimensional vector $(t_{i1}', t_{i2}', ..., t_{im}')$ where:
$t_{ij}' = (1+\epsilon)^{-(q-1)},$
$if\ (1+\epsilon)^{-q} < \frac{t_{ij}}{t_{max}} \leq (1+\epsilon)^{-(q-1)},\ q = 1,...,k$

In UM', let $\mathcal{E} = \{(1+\epsilon)^{-(q-1)},\ q = 1,...,k\}$ the set of all possible processing time values and $\mathcal{T}$ the set of possible task types. Clearly, for task i, we have $(t_{i1}', ..., t_{im}') \in \mathcal{E}^m$ and since $|\mathcal{E}| = k$: $|\mathcal{T}| \leq k^m$

Let b_l, $l \in \mathcal{T}$, the number of tasks belonging to task type l. Then, $\sum_{l \in \mathcal{T}} b_l = n$. If $x_{lj},\ l \in \mathcal{T},\ j = 1,..,m$, is a non negative integer indicating the number of tasks of type l assigned on processor j, we formulate UM':
$min\ \ C_{max}'\ \ s.t:\ \sum_{l \in \mathcal{T}} t_{lj}' x_{lj} \leq C_{max}',\ \ \ j = 1,..,m,$
$\sum_{j=1}^{m} x_{lj} = b_l,\ \ \ l \in \mathcal{T}\ \ \ and$

$x_{lj} \in N,\ l \in T,\ j = 1,..,m,\ C'_{max} \in R$

Working as in section 2, we prove the following:

Proposition 3 *When the number of processors m is fixed, UM' can be optimally solved in constant time.*

Proof: In UM', there are at most $k^m m$ assignment variables since $|T| \leq k^m$ (equation 4). Thus the maximum total number of variables in the formulation is $k^m m + 1$. Similarly, the number of constraints is $k^m + m$. For fixed m, both the number of variables and constraints are bounded above by constants and UM' can be optimally solved in constant time [6]. □

Proposition 4 *If $OPT(UM)$ and $OPT(UM')$ are the optimal solutions for UM and UM' respectively, we have:* $OPT(UM) \leq OPT(UM') \leq (1+\epsilon)\, OPT(UM)$

Proof: By the grouping procedure $t_{ij} \leq t'_{ij},\ \forall i = 1,...,n,\ \forall j = 1,..,m$ and the inequality $OPT(UM) \leq OPT(UM')$ still holds.

For the second part, we work in exactly the same way as in the proof of proposition 2 and we only give the sketch of the proof. If $FEAS(UM')$ is the solution obtained from $OPT(UM)$ when each t_{ij} is transformed into t'_{ij}, using the inequality:
$t'_{ij} < (1+\epsilon)\ t_{ij},\quad \forall i = 1,...,n,\ j = 1,...,m$, we bound the maximum distance between $FEAS(UM')$ and $OPT(UM)$: $FEAS(UM') < (1+\epsilon)\, OPT(UM)$. Since for the optimal solution of UM' we have: $OPT(UM') \leq FEAS(UM')$ the second part of the proposition is proved. □

The approximation scheme for scheduling unrelated machines on fixed number of processors is an extension of the approximation framework presented in section 2, where we consider a set of k^m distinct task types at most. Initially, we normalize task processing times into interval (0,1], dividing by $t_{max} = max\{t_{ij},\ i = 1,..,n,\ j = 1,..,m\}$, in $O(n)$ time. Next, for $k = \lceil \frac{-log\ t_{min}}{log(1+\epsilon)} \rceil$, we get UM' by grouping task processing times t'_{ij} into $\mathcal{E}$ and task types into T. The grouping procedure is performed in $O(n\ log\ (k^m)) = O(n\ m\ log\ \frac{-log\ t_{min}}{log(1+\epsilon)})$ time. Now, using proposition 3, we can find the optimal solution of UM' in constant time. Finally, we transform in linear time the determined solution into a feasible one for UM by changing t'_{ij} into initial t_{ij} and we obtain an ϵ-approximate solution for UM (proposition 4). The time complexity of the described approximation scheme is $O(n\ m\ log \frac{-log\ t_{min}}{log(1+\epsilon)})$.

5 Conclusion

In this paper, we have presented a general framework for approximation schemes for scheduling independent jobs on a fixed number of parallel processors. We have applied this general framework in the case of a fixed number of identical, uniform and unrelated processors. All resulting ϵ-approximation algorithms run in linear time on the number of jobs to be scheduled.

The interest of the approximation schemes presented for multiprocessor and uniform processor scheduling, lies on their asymptotically reduced time complexity. Furthermore, the approximation scheme for the problem of unrelated machines, improves the complexity of existing approximation schemes. The unfeasibility of an approximation scheme for the general problem [7] implies that the best result one can hope for scheduling unrelated machines is a fully polynomial time approximation scheme for fixed number of processors. In addition, the flexibility of the general approximation framework makes it suitable for other classes of important scheduling problems, such as resource constrained scheduling and typed task systems.

References

[1] W. Fernandez de la Wega and G. Lueker. Bin Packing can be Solved in 1+ϵ in Linear Time. *Combinatorica*, 1(4):349–355, 1981.

[2] D.S. Hochbaum and D.B. Shmoys. Using Dual Approximation Algorithms for Scheduling Problems: Theoritical and Practical Results. *JACM*, 34(1):144–162, January 1987.

[3] D.S. Hochbaum and D.B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, June 1988.

[4] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *JACM*, 23:317–327, 1976.

[5] N. Karmarkar and R. Karp. An Efficient Approximation Scheme for the One Dimentional Bin Packing problem. In *FOCS*, pages 312–320, 1982.

[6] H.W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[7] J.K. Lenstra, D.B. Schmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[8] J. Leung. Bin Packing with restricted piece sizes. *Information Processing Letters*, 31:145–149, 1989.

[9] S.K. Sahni. Algorithms for scheduling independent tasks. *JACM*, 23:117–127, 1976.