INSTITUT FÜR INFORMATIK

On optimality of exact and approximation algorithms for scheduling problems

Lin Chen, Klaus Jansen, Guochuan Zhang

Bericht Nr. 1303
June 5, 2013

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

On optimality of exact and approximation algorithms for scheduling problems

Lin Chen, Klaus Jansen, Guochuan Zhang

Bericht Nr. 1303
June 5, 2013

e-mail: chenlin198662@zju.edu.cn, kj@informatik.uni-kiel.de,
zgc@zju.edu.cn

On optimality of exact and approximation algorithms for scheduling problems*

Lin Chen[1] Klaus Jansen[2] Guochuan Zhang[1]

[1]College of Computer Science, Zhejiang University, Hangzhou, 310027, China

chenlin198662@zju.edu.cn, zgc@zju.edu.cn

[2] Department of Computer Science, Kiel University, 24098 Kiel, Germany

kj@informatik.uni-kiel.de

Abstract

We consider the classical scheduling problem on parallel identical machines to minimize the makespan. Under the exponential time hypothesis (ETH), lower bounds on the running times of exact and approximation algorithms are characterized. We achieve the following results: (1) For scheduling on a constant number m of identical machines, denoted by $Pm||C_{max}$, a fully polynomial time approximation scheme (FPTAS) of running time $(1/\epsilon)^{O(m^{1-\delta})}|I|^{O(1)}$ for any constant $\delta > 0$ implies that ETH fails (where $|I|$ is the length of the input). It follows that the best-known FPTAS of running time $O(n) + (m/\epsilon)^{O(m)}$ for the more general problem with a constant number m of unrelated machines $Rm||C_{max}$ is essentially the best possible. (2) For scheduling on an arbitrary number of identical machines, denoted by $P||C_{max}$, a polynomial time approximation scheme (PTAS) of running time $2^{O((1/\epsilon)^{1-\delta})}|I|^{O(1)}$ for any $\delta > 0$ also implies that ETH fails. Thus the best-known PTAS of running time $2^{O(1/\epsilon^2 \log^3(1/\epsilon))} + O(n \log n)$ is almost best possible in terms of running time. (3) For $P||C_{max}$, even if we restrict that there are n jobs and the processing time of each job is bounded by $O(n)$, an exact algorithm of running time $2^{O(n^{1-\delta})}$ for any $\delta > 0$ implies that ETH fails. Thus the traditional dynamic programming algorithm of running time $2^{O(n)}$ is essentially the best possible.

Keywords: Approximation schemes; Lower bounds; Exponential time hypothesis

*Part of the work was done when the first author was visiting Kiel University. Research was supported in part by Chinese Scholarship Council and NSFC (11271325).

1 Introduction

The theory of NP-hardness allows us to rule out polynomial time algorithms for many fundamental optimization problems under the complexity assumption $P \neq NP$. On the other hand, however, this does not give us (non-polynomial) lower bounds on the running time for such algorithms. For example, under the assumption $P \neq NP$, there could still be an algorithm with running time $n^{O(\log n)}$ for 3-SAT or bin packing. A stronger assumption, the Exponential Time Hypothesis (ETH), was introduced by Impagliazzo, Paturi, and Zane [10]:

Exponential Time Hypothesis (ETH): There is a positive real δ such that 3-SAT with n variables and m clauses cannot be solved in time $2^{\delta n}(n+m)^{O(1)}$.

Using the Sparsification Lemma by Impagliazzo et al. [10], the ETH assumption implies that there is no algorithm for 3-SAT with n variables and m clauses that runs in time $2^{\delta m}(n+m)^{O(1)}$ for a real $\delta > 0$ as well. Under the ETH assumption, lower bounds on the running time for several graph theoretical problems have been obtained via reductions between decision problems. For example, there is no $2^{\delta n}$ time algorithm for 3-Coloring, Independent Set, Vertex Cover, and Hamiltonian Path unless the ETH assumption fails. An essential property of the underlying strong reductions to show these lower bounds is that the main parameter, the number of vertices, is increased only linearly. These lower bounds together with matching optimal algorithms of running time $2^{O(n)}$ gives us some evidence that the ETH is true, i.e. that a subexponential time algorithm for 3-SAT is unlikely to exist. For a nice survey about lower bounds via the ETH we refer to Lokshtanov, Marx, and Saurabh [20]. Interestingly, using the ETH assumption one can also prove lower bounds on the running time of approximation schemes. For example, Marx [21] proved that there is no PTAS of running time $2^{O((1/\epsilon)^{1-\delta})}n^{O(1)}$ for Maximum Independent Set on planar graphs, unless the ETH fails.

There are only few lower bounds known for scheduling and packing problems. Chen et al. [3] showed that precedence constrained scheduling on m machines cannot be solved in time $f(m)|I|^{o(m)}$ (where $|I|$ is the length of the instance), unless the parameterized complexity class $W[1] = FPT$. Kulik and Shachnai [16] proved that there is no PTAS for the 2D knapsack problem with running time $f(\epsilon)|I|^{o(\sqrt{1/\epsilon})}$, unless all problems in SNP are solvable in sub-exponential time. Patrascu and Williams [25] proved using the ETH assumption a lower bound of $n^{o(k)}$ for sized subset sum with n items and cardinality value k. Recently, Jansen et al. [12] showed a lower bound of $2^{o(n)}|I|^{O(1)}$ for the subset sum and partition problem and proved that there is no PTAS for the multiple knapsack and 2D knapsack problem with running time $2^{o(1/\epsilon)}|I|^{O(1)}$ and $n^{o(1/\epsilon)}|I|^{O(1)}$, respectively.

In this paper, we consider the classical scheduling problem of jobs on identical machines with the objective of minimizing the makespan, i.e., the largest completion time. Formally, an instance I is given by a set $\mathcal{M}$ of m identical machines and a set $\mathcal{J}$ of n jobs with processing times p_j. The objective is to compute a non-preemptive schedule or an assignment $a : \mathcal{J} \to \mathcal{M}$ such that each job is executed by exactly one machine and the maximum load $\max_{i=1,\ldots,m} \sum_{j:a(j)=i} p_j$ among all machines is minimized. In scheduling theory, this problem is denoted by $Pm||C_{max}$ if m is a constant or $P||C_{max}$ if m is an arbitrary input.

This problem is NP-hard even if $m = 2$, and is strongly NP-hard if m is an input. On the other hand, for any $\epsilon > 0$ there is a $(1 + \epsilon)$-approximation algorithm for $Pm||C_{max}$ [9] and the more general problem $P||C_{max}$ [7]. Furthermore, there is a long history of improvements on the running time of such algorithms. We give a brief introduction as follows.

In 1976, Horowitz and Sahni [9] presented a fully polynomial-time approximation scheme (FPTAS) for a more generalized scheduling model $Rm||C_{max}$ on unrelated machines, where each job j could have different execution times p_{ij} on different machines. The running time of the algorithm in [9] is $O(nm(nm/\epsilon)^{m-1})$. Lenstra, Shmoys and Tardos [17] presented an alternative approximation scheme for the problem with running time $(n + 1)^{m/\epsilon}|I|^{O(1)}$. In 2001, Jansen and Porkolab [14] presented an FPTAS with running time $O(n(m/\epsilon)^{O(m)})$. This has been improved by Fishkin et al. [4] to $O(n) + (\log m/\epsilon)^{O(m^2)}$ and by Jansen and Mastrolilli [13] to $O(n) + (m/\epsilon)^{O(m)} = O(n) + (\log m/\epsilon)^{O(m \log m)}$. If ϵ is small enough (e.g. $\epsilon < 1/m$), the running time can be bounded by $O(n) + (1/\epsilon)^{O(m)}$ [13]. All the above mentioned algorithms are based on a combination of linear and dynamic programming and have a running time that depends exponentially on m. The running time increases drastically as m increases, which is also known as the curse of dimensionality. It is not yet known whether there is an FPTAS of running time $(1/\epsilon)^{o(m)}|I|^{O(1)}$ for $Pm||C_{max}$.

For $P||C_{max}$, where the number of machines is a part of input, Hochbaum and Shmoys [7] gave a polynomial-time approximation scheme (PTAS) of running time $(n/\epsilon)^{O(1/\epsilon^2)}$. It has been improved by Leung [19] to a running time of $(n/\epsilon)^{O(1/\epsilon \log(1/\epsilon))}$. In 1998, Alon et al. [1] and Hochbaum and Shmoys [8] presented an efficient polynomial-time approximation scheme (EPTAS) with running time $f(1/\epsilon)+O(n)$, where f is doubly exponential in $1/\epsilon$. Jansen [11] gave an EPTAS for the scheduling problem on identical machines $P||C_{max}$ and uniform machines $Q||C_{max}$ with running time $2^{O(1/\epsilon^2 \log^3(1/\epsilon))} + n^{O(1)}$. The algorithm is based on solving a mixed integer linear program (MILP). Recently, Jansen and Robenek [15] showed how to avoid the MILP and presented an algorithm with the same running time $2^{O(1/\epsilon^2 \log^3(1/\epsilon))} + n^{O(1)}$. Here the approximation scheme uses a combination of a linear program relaxation and a dynamic program. Interestingly, if the maximum distance $\|y^* - x^*\|_\infty$ between any optimum linear solution x^* and the closest optimum integer linear solution y^* is bounded by a polynomial in $1/\epsilon$, the running time can be bounded by $2^{O(1/\epsilon \log^2(1/\epsilon))} + n^{O(1)}$ [15].

As seen above there are a plenty of approximation schemes for both $Pm||C_{max}$ and $P||C_{max}$. A natural and important question is to find lower bounds on the running time for approximation schemes to solve the scheduling problem with the help of ETH.

Exact algorithms for the scheduling problem are also under extensive research. Recently Lente et al. [18] provided algorithms of running time $2^{n/2}$ and $3^{n/2}$ for $P2||C_{max}$ and $P3||C_{max}$, respectively. O'Neil [23, 24] gave a sub-exponential time algorithm of running time $2^{O(k\sqrt{|I|})}$ for the related bin packing problem where $|I|$ is the length of the input and k is the number of bins. Such an algorithm also works for the scheduling problem, and thus it is again natural to ask whether the sub-exponential time is the best possible.

The main contribution of this paper is to characterize lower bounds on the running times of exact and approximation algorithms for the classical scheduling problem. We prove the following theorems.

Theorem 1 *For any $\delta > 0$, there is no $2^{O((1/\epsilon)^{1-\delta})}\ |I|^{O(1)}$ time PTAS for $P||C_{max}$, unless ETH fails.*

Theorem 2 *For any $\delta > 0$, there is no $2^{O(n^{1-\delta})}$ time exact algorithm for $P||C_{max}$ with n jobs even if we restrict that the processing time of each job is bounded by $O(n)$, unless ETH fails.*

Theorem 3 *For any $\delta > 0$, there is no $(1/\epsilon)^{O(m^{1-\delta})}\ |I|^{O(1)}$ time FPTAS for $Pm||C_{max}$, unless ETH fails.*

Theorem 4 *For any $\delta > 0$, there is no $2^{O(m^{1/2-\delta}\sqrt{|I|})}$ time exact algorithm for $Pm||C_{max}$, unless ETH fails.*

We also prove the traditional dynamic programming algorithm for the scheduling problem actually runs in $2^{O(\sqrt{m|I|\log m}+m\log|I|)}$ time, and it is thus essentially the best exact algorithm in terms of running time. An overview about the known and new results for $P||C_{max}$ is listed in the Table 1:

Table 1: Lower and upper bounds on the running time

Algorithms	Upper bounds	Lower bounds
Approximation scheme	$2^{O(1/\epsilon^2\log^3(1/\epsilon))}+O(n\log n)$	$2^{O((1/\epsilon)^{1-\delta})}\ \lvert I\rvert^{O(1)}$
Approximation scheme	$O(n)+(1/\epsilon)^{O(m)}$ (if $\epsilon < 1/m$)	$(1/\epsilon)^{O(m^{1-\delta})}\ \lvert I\rvert^{O(1)}$
Exact algorithm	$2^{O(\sqrt{m\lvert I\rvert\log m}+m\log\lvert I\rvert)}$	$2^{O(m^{1/2-\delta}\sqrt{\lvert I\rvert})}$
Exact algorithm	$2^{O(n)}$	$2^{O(n^{1-\delta})}$ ($O(n)$ jobs and processing times)

Main Ideas in Designing a Reduction. Theorem 1 and Theorem 2 rely on a nearly linear reduction, which reduces the 3SAT problem with n clauses and at most $3n$ variables to the scheduling problem whose (optimal) makespan is bounded by $O(n^{1+\delta})$ for any $\delta > 0$.

The traditional reduction constructs a scheduling problem whose makespan is bounded by $O(n^{16})$ [5], and thus yields a lower bound of $2^{(1/\epsilon)^{1/16}}$. To improve it, the following idea is used. We construct jobs for variables and clauses, and try to represent the information 'variable z_i is in clause c_j' with the form of 'the two jobs corresponding to z_i and c_j are on the same machine'. We can assume that there is a huge job on each machine, leaving a gap if the load of each machine is required to be a specified value. If we define the processing times of the jobs corresponding to z_i and c_j to be i ($i \leq 3n$) and $4nj$ ($j \leq n$), and generate a gap of $4nj + i = O(n^2)$ (through a huge job), then this gap has to be filled up by the two jobs. To make the job processing times smaller, say, to $O(n^{3/2})$, the following two ideas are applied. One is that, if every clause c_j contains z_i, z_{i+1} and z_{i+2} such that $0 \leq i - j \leq O(1)$, then the two jobs for z_i and c_j could be defined to have processing times of i and $4n(i-j) + i = O(n)$, respectively. We create a gap of $4n(i-j) + 2i$. There are multiple ways of filling up such a gap by a variable and a clause job. However, we can prove that there is a unique way of filling up all such kind of gaps, in which a gap of $4n(i-j) + 2i$ is filled up by i and $4n(i-j) + i$.

The second idea assumes that there is a "proper partition" of the clauses so that they can be divided into $O(\sqrt{n})$ groups where each group contains $O(\sqrt{n})$ clauses, and every variable only appears in clauses of one group. Then if c_j is in group k, we re-index it as $j' \leq O(\sqrt{n})$ and re-index variable $z_i \in c_j$ as $i' \leq O(\sqrt{n})$. Let $x = O(\sqrt{n})$. The processing times of the jobs for z_i and c_j are defined as $kx^2 + i' = O(n^{3/2})$ and $kx^2 + j'x = O(n^{3/2})$. Again we can create a gap of $2kx^2 + j'x + i'$ and prove that it could only be filled up by the two jobs. Both ideas rely on a certain structure of the given 3SAT instance. However, we can use Tovey's method [26] to alter the instance so that its clauses could be divided into two subsets and we can apply one idea for one subset. It is possible to generalize the second idea to get even lower processing times by partitioning the set of clauses recursively, i.e., we first equally partition the clauses into n^δ groups, and then partition each group equally into n^δ subgroups, and so on, until each subgroup contains only n^δ clauses eventually. Basically, the partition process forms $1/\delta - 1$ levels from n^δ groups to $n^{1-\delta}$ groups each containing n^δ clauses, where the top level is denoted as level $1/\delta$ and the bottom level is level 2. For each clause c_j, it appears in a (unique) group at each level. Suppose it is in the k_i-th group for each level i, and it is the k_1-th element in the group at the bottom level. Then the job corresponding to c_j has a processing time of $k_{1/\delta}x^{1/\delta} + \cdots + k_2x^2 + k_1x = O(n^{1+\delta})$ where $x = O(n^\delta)$. The job for z_i is defined similarly as $k_{1/\delta}x^{1/\delta} + \cdots + k_2x^2 + k_1$.

Theorem 3 and Theorem 4 rely on a different reduction, which reduces the 3SAT problem with $O(n)$ variables and clauses to the scheduling problem on m machines whose makespan is bounded by $2^{O(n/m \log^{O(1)} m)}$.

The traditional reduction reduces 3SAT to 3DM (the 3-dimensional matching problem) with $q = O(n^2)$ elements, and then further reduces 3DM to the 2-machine scheduling problem with the makespan of $2^{O(q \log q)}$ [5]. To get a lower makespan, we generalize 3DM a bit to allow one-element and two-element matches (i.e., matches of the form (w_i) and (w_i, x_j)). We are able to reduce 3SAT to the generalized 3DM with $q = O(n)$ elements and matches. From 3DM to 2-machine scheduling, the traditional reduction lists all the elements. Suppose w_i is the $f(w_i)$-th element in the list. Then a job of processing time $\alpha^{f(w_i)} + \alpha^{f(x_j)} + \alpha^{f(y_k)}$ where $\alpha = q + 1$ is constructed for a match (w_i, x_j, y_k), and f is a sort of function to determine the index of an element. We create a gap of $\sum_{i=1}^{q} \alpha^i = (111 \cdots 11)_\alpha$ (through some huge job) on one machine, where this gap has to be filled up by a subset of jobs where every α^i term appears once, and they correspond to the 3-dimensional matching in which every element appears once. Let $|f|$ be the largest value returned by f. The makespan of the above reduction (from the generalized 3DM) is $2^{O(|f| \log n)} = 2^{O(n \log n)}$. To reduce the exponential term $O(|f| \log n)$ by utilizing the m-machine environment, we may allow $m - 1$ elements to share one 'bit', e.g., $f(w_{i_l}) = \lambda$ for $1 \leq l \leq m - 1$, and construct a job for (w_i, x_j, y_k) with processing time $g(w_i)\alpha^{f(w_i)} + g(x_j)\alpha^{f(x_j)} + g(y_k)\alpha^{f(y_k)}$ where $\alpha = m^{O(1)}$ and $g(w_i), g(x_j), g(y_k) < \alpha$. We create a gap similar as $(111 \cdots 11)_\alpha$ on $m - 1$ machines, and each gap should be filled up by a subset of jobs where every α^i term appears once. A 3-dimensional matching could thus be determined through the matches corresponding to the jobs on the $m - 1$ machines. The main difficulty of this idea is from (the generalized) 3DM to scheduling: f should be defined without the knowledge of the 3-dimensional matching, meanwhile it should ensure that given any 3-dimensional matching (if it exists), jobs corresponding to the matching admits a "proper partition", i.e., they could be partitioned and put onto $m - 1$ machines so that any two jobs sharing the same α^i term are on different machines. To handle this, we design f in a way such that the following partition is always a proper partition: given any 3-dimensional matching we always partition them into $m - 1$ groups such that matches containing the element $w_{kn/(m-1)+l}$ are in the same group for $0 \leq l \leq n/(m - 1) - 1$, and jobs corresponding to the matching are divided accordingly. We formulate the problem of designing the f satisfying the above property into a graph coloring problem and provide a greedy algorithm to solve it. The function f computed by the greedy algorithm satisfies $|f| = O(n/m \log m)$.

2 Scheduling on Arbitrary Number of Machines

We prove the following theorem in this section.

Theorem 5 *Assuming ETH, there is no $2^{O(K^{1-\delta})}|I_{sche}|^{O(1)}$ time algorithm which determines whether there is a feasible schedule of makespan no more than K for any $\delta > 0$.*

Given the above theorem, Theorem 1 follows directly. To see why, suppose Theorem 1 fails, then for some $\delta_0 > 0$ there exists a $2^{O((1/\epsilon)^{1-\delta_0})}\ |I|^{O(1)}$ time PTAS. We use this algorithm to test if there is a feasible schedule of makespan no more than K for the scheduling problem by taking $\epsilon = 1/(K + 1)$. If there exists a feasible schedule of makespan no more than K, then the PTAS returns a solution with makespan no more than $K(1 + \epsilon) < K + 1$. Otherwise, the makespan of the optimal solution is larger than or equal to $K + 1$, and the PTAS thus returns a solution at least $K + 1$. In a word, the

PTAS determines whether there is a feasible schedule of makespan no more than K in $2^{O((K+1)^{1-\delta_0})}\,|I|^{O(1)}$ time, which is a contradiction to Theorem 5.

To prove Theorem 5, we start with a modified version of the 3SAT problem, say, 3SAT' problem, in which the set of clauses could be divided into sets C_1 and C_2 such that

- Every clause of C_1 contains three variables; every variable appears once in clauses of C_1
- Every clause of C_2 is of the form $(z_i \vee \neg z_k)$; every positive (negative) literal appears once in C_2
- If a 3SAT' instance is satisfiable, every clause of C_2 is satisfied by exactly one literal

There is a reduction from 3SAT (with m clauses) to 3SAT' via Tovey's method [26] which only increases the number of clauses and variables by $O(m)$, and thus ensures the following lemma.

Lemma 1 *Assuming ETH, there exists some $s > 0$ such that there is no 2^{sn} time algorithm for the 3SAT' problem with n variables.*

Proof. Given any instance I_{sat} (with m clauses) of the 3SAT problem, we may transform I_{sat} into a 3SAT' instance I'_{sat} in which every variable appears exactly three times. Such a transformation is due to Tovey and we describe it as follows for the completeness.

Let z be any variable in I_{sat} and suppose it appears d times in clauses. If $d = 1$ then we add a dummy clause $(z \vee \neg z)$. Otherwise $d \geq 2$ and we introduce d new variables $z_1, z_2, \cdots, z_d$ and d new clauses $(z_1 \vee \neg z_2)$, $(z_2 \vee \neg z_3)$, $\cdots$, $(z_d \vee \neg z_1)$. Meanwhile we replace the d occurrences of z by $z_1, z_2, \cdots, z_d$ in turn and remove z. By doing so we transform I_{sat} into I'_{sat} by introducing at most $3m$ new variables and $3m$ new clauses. Notice that each new clause we add is of the form $(z_i \vee \neg z_k)$. We let C_2 be the set of them and let C_1 be the set of other clauses. It is not difficult to verify that I'_{sat} is an instance of 3SAT', and is satisfiable if and only if I_{sat} is satisfiable. According to ETH, the lemma follows directly. □

From now on we use I_{sat} to denote a 3SAT' instance with n variables. Notice that the special structure of a 3SAT' instance implies that n could be divided by 3, and clauses could be divided into C_1 and C_2 such that $|C_1| = n/3$, $|C_2| = n$, and every variable appears once in clauses of C_1. We can re-index all the variables so that every clause $c_i \in C_1$ contains variables z_i, z_{i+1} and z_{i+2} for $i \in R = \{1, 4, 7, \cdots, n-2\}$.

Given any $\delta > 0$ (where $1/\delta \geq 2$ is a constant integer), we may further assume that n is sufficiently large (e.g., $n \geq 2^{3/\delta^2+7/\delta}$) and n^δ is an integer. (Indeed, if n^δ is not an integer, we can simply choose $\sigma \in \{0, 1, 2\}$ such that $\lceil n^\delta \rceil + \sigma$ could be divided by 3 and add $(\lceil n^\delta \rceil + \sigma)^{1/\delta} - n = 3\rho$ dummy variables into I'_{sat}. Let z_1 to $z_{3\rho}$ be these dummy variables. We add ρ dummy clauses $(z_1 \vee z_2 \vee z_3)$, $\cdots$, $(z_{3\rho-2} \vee z_{3\rho-1} \vee z_{3\rho})$ into C_1 and 3ρ

dummy clauses $(z_1 \vee \neg z_1)$, $\cdots$, $(z_{3\rho} \vee \neg z_{3\rho})$ into C_2. Obviously these dummy variables and clauses do not change the satisfiability of I'_{3sat}. Furthermore, given the fact that $e^\delta \geq 1+\delta$, $\lceil n^\delta \rceil + \sigma \leq n^\delta + \delta n^\delta \leq (en)^\delta$, thus $3\rho \leq (e-1)n$, which means that we add at most $2n$ dummy variables and clauses.)

In the following part of this section, we will construct a scheduling instance with $O(n/\delta)$ jobs and $O(n/\delta)$ machines such that it admits a feasible solution with makespan no more than $K = O(2^{3/\delta} n^{1+\delta})$ if and only if I_{sat} is satisfiable. This would be enough to prove Theorem 5, to see why, suppose the theorem fails, then an exact algorithm of running time $2^{O(K^{1-\delta_0})}|I_{sche}|^{O(1)}$ exists for some $\delta_0 > 0$. We take $\delta = \delta_0$ in the reduction, and we can determine in $2^{O(n^{1-\delta_0^2})}|I|^{O(1)} = 2^{o(n)}$ time whether the constructed scheduling instance admits a schedule of makespan K, and thus determine whether the given 3SAT' instance is satisfiable, which is a contradiction.

To provide an overview of the whole construction and the proof, we give a simplified reduction where $\delta = 1/2$ in the next subsection. We give the reduction for arbitrary δ after the simplified reduction.

2.1 The reduction for $\delta = 1/2$

We give a brief overview. For each positive (negative) literal, say, z_i (or $\neg z_i$), two pairs of jobs $v^\gamma_{i,1}$ and $v^\gamma_{i,2}$ ($v^\gamma_{i,3}$ and $v^\gamma_{i,4}$) are constructed where $\gamma \in \{T, F\}$. For each clause of C_1, say, c_j, one job u^T_j and two copies of job u^F_j are constructed.

As we have mentioned, we create huge jobs so as to make a gap on every machine. There are five kinds of huge jobs (gaps).

[**1.**] Variable-assignment gaps. To fill up these gaps either $v^F_{i,1}$, $v^F_{i,2}$, $v^T_{i,3}$, $v^T_{i,4}$ (meaning variable z_i is true), or $v^T_{i,1}$, $v^T_{i,2}$, $v^F_{i,3}$, $v^F_{i,4}$ (meaning variable z_i is false) are used. Jobs a^γ_i, b^γ_i, c^γ_i and d^γ_i are created as 'assistant jobs' to help achieve the above requirement.

[**2.**] Variable-clause gaps. If the positive (or negative) literal z_i (or $\neg z_i$) is in $c_j \in C_1$, then a variable-clause gap is created so that it could only be filled up by u_j and $v_{i,1}$ (or $v_{i,3}$). Notice that $i = j, j+1, j+2$, the first idea in the introduction is used to ensure that the gap can only be filled up in this way. Furthermore, for the superscripts of u_j and $v_{i,1}$ (or $v_{i,3}$), the gap enforces that only three combinations are valid: (T,T), (F,F) and (F,T). Recall that there is one u^T_j, it is thus always scheduled with a true variable job, say, $v^T_{i',1}$ (or $v^T_{i',3}$), indicating that the clause is satisfied by $z_{i'}$ (or $\neg z_{i'}$).

[**3.**] Variable-agent and agent-agent gaps. We want to create a gap for $(z_i \vee \neg z_k) \in C_2$ so that it could only be filled up by $v^T_{i,2}$ and $v^F_{k,4}$ (or $v^F_{i,2}$ and $v^T_{k,4}$), indicating that $(z_i \vee \neg z_k)$ is satisfied by z_i (or $\neg z_k$). However, such a gap could not be constructed directly since the the size of a gap should be $O(n^{3/2})$. We use the following idea to achieve this. We create a pair of agent jobs for $v_{i,2}$ (or $v_{k,4}$), namely $\eta^\gamma_{i,+}$ (or $\eta^\gamma_{i,-}$). We create one variable-agent gap which could only be filled up by $v_{i,2}$ and its agent $\eta_{i,+}$, and they should be one true and one false (i.e., their superscripts are T and F). Similarly another variable-agent gap is created which could only be filled up by $v_{k,4}$ and $\eta_{k,-}$ that are one true and one false.

We further create an agent-agent gap which could only be filled up by $\eta_{i,+}$ and $\eta_{k,-}$ that are one true and one false. Combining the three gaps, we can conclude that the $v_{i,2}$ and $v_{k,4}$ used in these gaps are one true and one false. The second idea in the introduction is used to construct these gaps.

[4.] Variable-dummy gaps. Recall that we construct 8 jobs for a variable and only use 7 of them (either $v_{i,1}$ or $v_{i,3}$ is left), the remaining one will be used to fill these gaps.

2.1.1 Partition Clauses

Given a 3SAT' instance I_{sat} with n variables, we know that $C_1 = n/3$ and $C_2 = n$. We may further assume that n is sufficiently large (i.e., $n \geq 2^{26}$) and $\sqrt{n}$ is an integer.

Recall that there are n clauses in C_2 and every positive (negative) literal appears once in them. We partition clauses of C_2 equally into $\sqrt{n}$ groups. Let $S_n = \{1, 2, \cdots, n\}$. We define the function $f : S_n \to S_{\sqrt{n}}$ such that the positive literal z_i is in group $f(i)$, and we define the function $\bar{f} : S_n \to S_{\sqrt{n}}$ such that the negative literal $\neg z_i$ is in group $\bar{f}(i)$.

In each group, say, group i, there are $\sqrt{n}$ different positive literals. Let their indices be $i_1 < i_2 < \cdots < i_{\sqrt{n}}$, then we define $g : S_n \to S_{\sqrt{n}}$ such that $g(i_k) = k$. Similarly the indices of negative literals could be listed as $\bar{i}_1 < \bar{i}_2 < \cdots < \bar{i}_{\sqrt{n}}$ and we define $\bar{g} : S_n \to S_{\sqrt{n}}$ such that $\bar{g}(\bar{i}_k) = k$.

Our definition of g and $\bar{g}$ implies the following lemma.

Lemma 2 *For any $i, i' \in S_n$ and $i < i'$, if $f(i) = f(i')$, then $g(i) < g(i')$. Similarly if $\bar{f}(i) = \bar{f}(i')$, then $\bar{g}(i) < \bar{g}(i')$.*

Furthermore, if $(z_i \vee \neg z_k) \in C_2$, then $f(i) = \bar{f}(k)$ according to our definition.

2.1.2 Construction of the Scheduling Instance

Given I_{sat}, we construct an instance of scheduling problem with $30n$ jobs and $9n$ machines, and prove that I_{sat} is satisfiable if and only if there exists a feasible solution for the constructed scheduling instance with makespan no more than $K = 10^5 r$, where $r = 2^{15} n^{3/2}$. Throughout this section we set $x = 4\sqrt{n}$ and use $s(j)$ to denote the processing time of job j. $\gamma \in \{T, F\}$.

$20n$ jobs are constructed for variables, among them there are $8n$ variable jobs, $4n$ agent jobs and $8n$ truth assignment jobs. n jobs are constructed for clauses of C_1. $9n$ huge jobs are constructed so that there is one on each machine.

Variable jobs: $v^\gamma_{i,1}$ and $v^\gamma_{i,2}$ are constructed for z_i, $v^\gamma_{i,3}$ and $v^\gamma_{i,4}$ are for $\neg z_i$.

$$s(v^T_{i,k}) = r + 2^9(f(i)x^2 + i) + 2^8 + k, \quad k = 1, 2$$

$$s(v^T_{i,k}) = r + 2^9(\bar{f}(i)x^2 + i) + 2^8 + k, \quad k = 3, 4$$

$$s(v^F_{i,k}) = s(v^T_{i,k}) + 2r, \quad k = 1, 2, 3, 4$$

Agent jobs: $\eta_{i,+}^{\gamma}$ and $\eta_{i,-}^{\gamma}$.

$$s(\eta_{i,+}^{T}) = r + 2^9(f(i)x^2 + g(i)) + 2^7 + 8,$$

$$s(\eta_{i,-}^{T}) = r + 2^9(\bar{f}(i)x^2 + \bar{g}(i)x) + 2^7 + 16,$$

$$s(\eta_{i,\sigma}^{F}) = s(\eta_{i,\sigma}^{T}) + 2r, \quad \sigma = +, -$$

Truth assignment jobs: a_i^{γ}, b_i^{γ}, c_i^{γ} and d_i^{γ}.

$$s(a_i^F) = 11r + (2^7 i + 8), s(b_i^F) = 11r + (2^7 i + 32),$$

$$s(c_i^F) = 101r + (2^7 i + 16), s(d_i^F) = 101r + (2^7 i + 64),$$

$$s(k_i^T) = s(k_i^F) + r, \quad k = a, b, c, d.$$

Clause jobs: 3 clause jobs are constructed for every $c_j \in C_1$ where $j \in R$, with one u_j^T and two copies of u_j^F:

$$s(u_j^T) = 10004r + 2^{11}j, s(u_j^F) = 10002r + 2^{11}j.$$

Dummy jobs: $n + n/3$ jobs with processing time $1000r$, and $n - n/3$ jobs with processing time $1002r$.

Let V and V_a be the set of variable jobs and agent jobs. Let A, B, C, D be the set of a_i^{γ}, b_i^{γ}, c_i^{γ} and d_i^{γ} respectively. Sometimes we may drop the superscript for simplicity, e.g., we use a_i to represent a_i^T or a_i^F.

We construct huge jobs. There are five kinds of huge jobs corresponding to the five kinds of gaps we mention before.

Two huge jobs (variable-agent jobs) $\theta_{\eta,i,+}$ and $\theta_{\eta,i,-}$ are constructed for each variable z_i:

$$\begin{aligned} s(\theta_{\eta,i,+}) &= 10^5 r - 4r - 2^9[2f(i)x^2 + g(i) + i] - (2^8 + 2^7 + 10) \\ s(\theta_{\eta,i,-}) &= 10^5 r - 4r - 2^9[2\bar{f}(i)x^2 + \bar{g}(i)x + i] - (2^8 + 2^7 + 20) \end{aligned}$$

One huge job (agent-agent job) θ_{i,k,C_2} is constructed for $(z_i \vee \neg z_k) \in C_2$:

$$s(\theta_{i,k,C_2}) = 10^5 r - 4r - 2^9[f(i)x^2 + \bar{f}(k)x^2 + \bar{g}(k)x + g(i)] + 2^8 + 24].$$

Notice that $f(i) = \bar{f}(k)$ according to our definition of f and $\bar{f}$.

Three huge jobs (variable-clause jobs) are constructed for each $c_j \in C_1$ ($j \in R$), one for each literal: for $i = j, j+1, j+2$, if $z_i \in c_j$, we construct $\theta_{j,i,+,C_1}$, otherwise $\neg z_i \in c_j$, and we construct $\theta_{j,i,-,C_1}$.

$$s(\theta_{j,i,+,C_1}) = 10^5 r - 11005r - (2^9 f(i)x^2 + 2^{11}j + 2^9 i + 2^8 + 1),$$

$$s(\theta_{j,i,-,C_1}) = 10^5 r - 11005r - (2^9 \bar{f}(i)x^2 + 2^{11}j + 2^9 i + 2^8 + 3).$$

One huge job (variable-dummy job) is constructed for each variable. Notice that each variable appears exactly three times in clauses, if z_i appears twice while $\neg z_i$ appears once, we construct $\theta_{i,-}$. Otherwise, we construct $\theta_{i,+}$ instead.

$$s(\theta_{i,+}) = 10^5r - 1003r - (2^9f(i)x^2 + 2^9i + 2^8 + 1),$$

$$s(\theta_{i,-}) = 10^5r - 1003r - (2^9\bar{f}(i)x^2 + 2^9i + 2^8 + 3).$$

Thus, for each clause c_j ($j \in R$) and $i = j, j+1, j+2$, either $\theta_{i,+}$ and $\theta_{j,i,-,C_1}$ exist, or $\theta_{i,-}$ and $\theta_{j,i,+,C_1}$ exist.

Four huge jobs (variable-assignment jobs) are constructed for each variable z_i, namely $\theta_{i,a,c}$, $\theta_{i,b,d}$, $\theta_{i,a,d}$ and $\theta_{i,b,c}$:

$$s(\theta_{i,a,c}) = 10^5r - 115r - 2^9(f(i)x^2 + i) - (2^8 + 2^8i + 25),$$

$$s(\theta_{i,b,d}) = 10^5r - 115r - 2^9(f(i)x^2 + i) - (2^8 + 2^8i + 98),$$
$$s(\theta_{i,a,d}) = 10^5r - 115r - 2^9(\bar{f}(i)x^2 + i) - (2^8 + 2^8i + 75),$$
$$s(\theta_{i,b,c}) = 10^5r - 115r - 2^9(\bar{f}(i)x^2 + i) - (2^8 + 2^8i + 52).$$

It is not difficult to verify that the total processing time of all the jobs is $9n \cdot 10^5r$. Furthermore, if the given 3SAT' instance I_{sat} is satisfiable, then the constructed scheduling instance I_{sche} admits a feasible schedule whose makespan is 10^5r (the reader may refer to Subsection 2.2.3 for a similar proof).

2.1.3 Scheduling to 3SAT

We prove that if there is a schedule whose makespan is no more than 10^5r (which implies that the load of each machine is exactly 10^5r), then I_{sat} is satisfiable.

The input of a scheduling instance is a set of integers with $9n$ identical machines, we prove that we can determine the symbol (e.g., a_i^T, u_j^F) of a job based on its processing time. Recall that we define the processing time of a job in the form of a polynomial, which could be partitioned into four terms, the r-term, x^2-term, x-term and constant term (the summation of all terms without r or x). The sum of the x-term and constant term is called small-x^2-term, and the sum of x^2-term, x-term and constant term is called small-r-term. Since $x = 4\sqrt{n}$ and $r = 2^{15}n^{3/2}$, there are gaps between terms.

Lemma 3 *The small-r-term and small-x^2-term of a huge job are negative with their absolute values bounded by $1/2r$ and $2^9 \cdot 3/4x^2$ respectively. The small-r-term of any other job is positive and bounded by $1/4r$. The small-x^2-term of a variable or agent job is positive and bounded by $2^9 \cdot 3/8x^2$.*

The reader may refer to Lemma 8 for a similar proof. The above lemma allows us to determine the r-term and x^2-term of a job, and the symbol of it could be determined easily if it is a variable, agent, truth assignment, clause or dummy job. If it is a huge job, we observe that, the function g is defined in the way that if $f(i_1) = f(i_2)$ for $i_1 < i_2$, then $g(i_1) < g(i_2)$ (Lemma 2), which implies that $i_1 + g(i_1) < i_2 + g(i_2)$, thus the processing time of a huge job is unique and we can also determine its symbol.

Let Sol^* be an optimal solution, it can be easily seen that there is a huge job on each machine, creating a gap. The following lemmas follows by considering the r-terms and the residuals of dividing each job by 2^7 (the reader may refer to Lemma 10 for a similar proof).

Lemma 4 *The following statements hold.*

- *A variable-agent gap is filled up with a variable job and an agent job.*
- *An agent-agent gap is filled up with two agent jobs.*
- *A variable-clause gap is filled up with a clause job, a variable job and a dummy job.*
- *A variable-dummy gap is filled up with a variable job and a dummy job.*
- *A variable-assignment gap is filled up with a variable job and two truth-assignment jobs, one in $A \cup B$, the other in $C \cup D$.*

Combining Lemmas 3 and 4, we have the following lemma.

Lemma 5 *For jobs on each machine, their r-terms add up to $10^5 r$, x^2-terms and small-x^2-terms add up to 0.*

Consider the x^2-terms of gaps. An agent-agent gap or variable-agent gap is called a regular gap, since their x^2-terms are $2^9 \cdot 2\zeta x^2$ where $1 \leq \zeta \leq \sqrt{n}$. Other gaps are called singular gaps with the x^2-terms being $2^9 \zeta x^2$.

A singular gap is called well-canceled, if it is filled up by jobs (other than huge jobs) whose x^2-terms are $2^9 \zeta x^2$ and 0. A regular gap is called well-canceled, if it is filled up by two jobs whose x^2-terms are both $2^9 \zeta x^2$.

Lemma 6 *Every singular gap is well-canceled, and every regular gap is well-canceled.*

Proof. We briefly argue why it is the case. The first part follows directly from Lemma 4 and Lemma 5. We show the second part.

Consider the regular gap with the term $2^9 \cdot 2x^2$. Since it is filled up by two variable or agent jobs (Lemma 4), whose x^2-term is at least $2^9 x^2$, thus it is obviously well-canceled.

According to the construction of f and $\bar{f}$, there are $\sqrt{n}$ indices such that $f(i) = 1$ and $\sqrt{n}$ indices such that $\bar{f}(i) = 1$, thus in all there are $12\sqrt{n}$ variable and agent jobs with the term $2^9 x^2$. There are $\sqrt{n}$ variable-clause gaps, $\sqrt{n}$ variable-dummy gaps and $4\sqrt{n}$ variable-assignment gaps with the term $2^9 x^2$, meanwhile there are $2\sqrt{n}$ variable-agent gaps and $\sqrt{n}$ agent-agent gaps with the term $2^9 \cdot 2x^2$. All of these gaps are well-canceled, implying that all the variable and agent jobs with 2^9x^2 are used to fill up these gaps. Thus to fill up a regular gap with the term of $2^9 \cdot 4x^2$, we have to use variable or agent jobs with the term of at least $2^9 \cdot 2x^2$, implying that this regular gap is also well-canceled. Iteratively applying the above arguments, every regular gap is well-canceled. □

A huge job (gap) is called satisfied, if the indices of other jobs on the same machine with it coincide with its index. For example, the variable-clause job $\theta_{j,i,-,C_1}$ is satisfied if it is on the same machine with the variable job $v_{i,k}$ and clause job u_j where $k \in \{1, 2, 3, 4\}$, and $\theta_{i,a,c}$ is satisfied if it is with $v_{i,k}$, a_i and c_i.

Lemma 7 *Every huge job (gap) is satisfied.*

Proof. We give the sketch of proof. It is easy to see that every variable-dummy job is satisfied. According to the definition of f and g (f' and g'), an index i is determined uniquely by the pair $(f(i), g(i))$ (or $(\bar{f}(i), \bar{g}(i))$). Combining this fact with Lemma 6, it is not difficult to verify that every agent-agent job θ_{i,k,C_2} is scheduled with $\eta_{i,\sigma}$ and $\eta_{k,\sigma'}$ where $\sigma, \sigma' \in \{+, -\}$, and is thus satisfied.

Consider the variable-agent job $\theta_{\eta,1,+}$. According to Lemma 4 and Lemma 6, the gap of $4r + 2^9(2f(1)x^2 + g(1) + 1) + 2^8 + 2^7 + 10$ should be filled up by a variable job $v_{i',k}$ and an agent job $\eta_{i'',\sigma}$ where $k \in \{1, 2, 3, 4\}$ and $\sigma \in \{+, -\}$, such that $f(i') = f(i'') = 1$. Simple calculations show that $g(i'') + i' = g(1) + 1$. Since $i, i' \geq 1$, Lemma 2 implies that $i' = i'' = 1$, and $\theta_{\eta,1,+}$ is thus satisfied. Similarly we can prove that $\theta_{\eta,1,-}$ is satisfied.

Using similar arguments, it is not difficult to verify that the three variable-clause job $\theta_{1,i,\sigma_i,C_1}$ ($\sigma_i \in \{+, -\}$ for $i = 1, 2, 3$) and the four variable-assignment jobs $\theta_{1,a,c}$, $\theta_{1,b,d}$, $\theta_{1,a,d}$, $\theta_{1,b,c}$ are satisfied. We call $v_{i,k}$ ($k \in \{1, 2, 3, 4\}$) and $\eta_{i,\sigma}$ ($\sigma \in \{+, -\}$) as jobs of index-level i, then all the jobs of index-level 1 are used to fill up the the previous mentioned gaps so that when we consider $\theta_{\eta,2,+}$, it should be scheduled together with $v_{i',k}$ and $\eta_{i'',\sigma}$ with $i', i'' \geq 2$, and we can carry on the previous arguments. □

The reader may refer to Lemma 17 for a similar proof. With the above lemma, it is not difficult to further verify (due to the residuals of each job divided by 2^7) that jobs are scheduled according to the following table. Recall that for every $j \in R$ and $i = j, j + 1, j + 2$, either $\theta_{j,i,+,C_1}$ and $\theta_{i,-}$ exist, or $\theta_{j,i,-,C_1}$ and $\theta_{i,+}$ exist.

Table 2: Indices of jobs

$\theta_{i,a,c}$	$v_{i,1}$	$\theta_{j,i,+,C_1}$	$v_{i,1}$	$\theta_{i,+}$	$v_{i,1}$
$\theta_{i,b,d}$	$v_{i,2}$	$\theta_{j,i,-,C_1}$	$v_{i,3}$	$\theta_{i,-}$	$v_{i,3}$
$\theta_{i,a,d}$	$v_{i,3}$	$\theta_{\eta,i,+}$	$v_{i,2}$		
$\theta_{i,b,c}$	$v_{i,4}$	$\theta_{\eta,i,-}$	$v_{i,4}$		

The previous discussion determines the indices of jobs on each machine, and we need to further determine their superscripts. We have the following simple observations (by considering the r-terms of jobs).

- The two jobs with an agent-agent or variable-agent job are one true and one false.
- The three jobs with a variable-assignment job are either (T,T,T) or (F,F,F).
- The clause job and variable job with a variable-clause job are (T,T), (F,F) or (F,T), i.e., the variable job must be true if the clause job is true.

According to the superscripts of jobs on each machine, we give a truth assignment of variables in the following way. Notice that according to the analysis above, there are two way of scheduling truth-assignment jobs, either $(v_{i,1}^T, a_i^T, c_i^T)$, $(v_{i,2}^T, b_i^T, d_i^T)$, $(v_{i,3}^F, a_i^F, d_i^F)$, $(v_{i,4}^F, b_i^F, c_i^F)$ or $(v_{i,1}^F, a_i^F, c_i^F)$, $(v_{i,2}^F, b_i^F, d_i^F)$, $(v_{i,3}^T, a_i^T, d_i^T)$, $(v_{i,4}^T, b_i^T, c_i^T)$. We let variable z_i be false if a_i^T is with $v_{i,1}^T$, otherwise it is true. We prove that I_{sat} is satisfied.

Consider any $c_j \in C_1$, u_j^T should be scheduled with a true variable job and it is either $v_{i,1}^T$ or $v_{i,3}^T$. If it is $v_{i,1}^T$, then obviously the variable z_i is true (for otherwise $v_{i,1}^T$ is with a_i^T, rather than u_j^T). Furthermore, u_j^T and $v_{i,1}^T$ must be scheduled with $\theta_{j,i,+,C_1}$. Recall that we construct the job $\theta_{j,i,+,C_1}$ if the positive literal $z_i \in c_j$. Thus c_j is satisfied. Otherwise it is $v_{i,3}^T$, and $v_{i,3}^F$ is with a_i^F, meaning that a_i^T is with $v_{i,1}^T$ and the variable z_i is false. Similar arguments show that $\neg z_i \in c_j$ and c_j is also satisfied.

Consider any $(z_i \vee \neg z_k) \in C_2$. θ_{i,k,C_2} is scheduled with two agent jobs, one true and one false. If it is with $\eta_{i,+}^T$, then $\eta_{i,+}^F$ is with $v_{i,2}^T$, implying that $v_{i,2}^F$ is with b_i^F and d_i^F. Hence the variable z_i is true, meaning that $(z_i \vee \neg z_k)$ is satisfied. Otherwise θ_{i,k,C_2} is with $\eta_{i,+}^F$, and using similar arguments we can show that $(z_i \vee \neg z_k)$ is also satisfied.

Remark The reader can see that, the x^2-terms of variable jobs are only used when we try to prove that variable-agent and agent-agent jobs are satisfied. Indeed, the satisfaction of $(z_i \vee \neg z_k) \in C_2$ follows from $v_{i,2}$ and $v_{k,4}$ should be one true and one false (meaning that the positive literal z_i and negative literal $\neg z_k$ should be one true and one false), and this is ensured by the following two facts.

- $v_{i,2}$ ($v_{k,4}$) has two agents, one true and one false. $v_{i,2}$ ($v_{k,4}$) is scheduled with one of its agents, and they are one true and one false.
- One agent of $v_{i,2}$ is scheduled together with one agent of $v_{k,4}$, and they are one true and one false.

The processing time of an agent job should be defined in a proper way so that we can determine from the gaps that a variable job is scheduled with its corresponding agent job, and two specific agent jobs are scheduled together, and this requires a processing time of $O(n^{3/2})$. To reduce it to $O(n^{1+\delta})$ for $\delta > 0$, we create $1/\delta - 1$ pairs of agent jobs (from layer-1 to layer-$(1/\delta - 1)$) for a variable. A variable job is scheduled with its layer-$(1/\delta - 1)$ agent job, its layer-$(1/\delta - 1)$ agent job is with its layer-$(1/\delta - 2)$ agent job, $\cdots$, its layer-2 agent job is with its layer-1 agent job, and two specific layer-1 agent jobs are scheduled together. Detailed description is in the next subsection.

2.2 Reduction for arbitrary δ

We give a brief overview. We create jobs $v_{i,1}^{\gamma}$, $v_{i,2}^{\gamma}$, $v_{i,3}^{\gamma}$, $v_{i,4}^{\gamma}$ for variable z_i and jobs u_j^{γ} for $c_j \in C_1$ where $\gamma \in \{T, F\}$, just as what we do when $\delta = 1/2$. Recall that in the previous subsection we create 2 pairs of agent jobs $\eta_{i,+}^{\gamma}$ and $\eta_{i,-}^{\gamma}$ for every variable z_i, in this subsection we create $2(1/\delta - 1)$ pairs of agent jobs, namely $\eta_{i,j,+}^{\gamma}$ and $\eta_{i,j,-}^{\gamma}$ for $1 \le j \le 1/\delta - 1$. For simplicity, $\eta_{i,j,+}^{\gamma}$ and $\eta_{i,j,-}^{\gamma}$ are called layer-j agent jobs. Again we create huge jobs so as to make a gap on every machine. There are six kinds of huge jobs (gaps).

[**1.**] Variable-assignment gaps. Similar as the previous subsection, to fill up these gaps either $v_{i,1}^{F}$, $v_{i,2}^{F}$, $v_{i,3}^{T}$, $v_{i,4}^{T}$ (meaning variable z_i is true), or $v_{i,1}^{T}$, $v_{i,2}^{T}$, $v_{i,3}^{F}$, $v_{i,4}^{F}$ (meaning variable z_i is false) are used. Jobs a_i^{γ}, b_i^{γ}, c_i^{γ} and d_i^{γ} also serve as 'assistant jobs'.

[**2.**] Variable-clause gaps. Similar as the previous subsection, if the positive (or negative) literal z_i (or $\neg z_i$) is in $c_j \in C_1$, then a variable-clause gap is created so that it could only be filled up by u_j and $v_{i,1}$ (or $v_{i,3}$), and their superscripts can only be (T,T), (F,F) or (F,T).

[**3.**] Variable-agent, layer-decreasing and agent-agent gaps. Similar as what we do in the previous subsection, we try to construct several gaps so as to enforce that either $v_{i,2}^{T}$, $v_{k,4}^{F}$ or $v_{i,2}^{F}$, $v_{k,4}^{T}$ are are used to fill up these gaps. We create $1/\delta - 1$ pairs of agent jobs for $v_{i,2}$ (or $v_{k,4}$), namely $\eta_{i,j,+}^{\gamma}$ (or $\eta_{i,j,-}^{\gamma}$) for $1 \le j \le 1/\delta - 1$. We create one variable-agent gap which could only be filled up by $v_{i,2}$ and the corresponding layer-$(1/\delta - 1)$ agent job $\eta_{i,1/\delta-1,+}$, and they should be one true and one false (i.e., their superscripts are T and F). We then create $1/\delta - 2$ layer-decreasing gaps such that the j-th gap could only be filled up by $\eta_{i,j+1,-}$ and $\eta_{i,j,-}$ that are one true and one false. Similarly another variable-agent gap is created which could only be filled up by $v_{k,4}$ and $\eta_{k,1/\delta-1,-}$ that are one true and one false, and another $1/\delta - 2$ layer-decreasing gaps are created such that the j-th gap could only be filled up by $\eta_{i,j+1,+}$ and $\eta_{i,j,+}$ that are one true and one false. Finally we create an agent-agent gap which could only be filled up by $\eta_{i,1,+}$ and $\eta_{k,1,-}$ that are one true and one false. It is not difficult to verify that if all these gaps are filled up, then either $v_{i,2}^{T}$, $v_{k,4}^{F}$ or $v_{i,2}^{F}$, $v_{k,4}^{T}$ are are used together with all the agent jobs.

[**4.**] Variable-dummy gaps. Again we construct 8 jobs for a variable and only use 7 of them (either $v_{i,1}$ or $v_{i,3}$ is left), the remaining one will be used to fill these gaps.

2.2.1 Partition Clauses

Recall that n^{δ} is an integer. We first partition all the clauses (of C_2) equally into n^{δ} groups. Let these groups be S_{1,k_1} for $1 \le k_1 \le n^{\delta}$. We call them as layer-1 groups.

It can be easily seen that each layer-1 group contains exactly $n_1 = n^{1-\delta}$ clauses, as a consequence, clauses of S_{1,k_1} contain n_1 positive literals and n_1 negative literals.

For simplicity, let S^+_{1,k_1} be the indices of all the positive literals of S_{1,k_1} and S^-_{1,k_1} be the indices of all the negative literals of S_{1,k_1}.

Suppose $i_1^{(1,k_1)} < i_2^{(1,k_1)} < \cdots < i_{n_1}^{(1,k_1)}$ are all the indices in S^+_{1,k_1}, we then define

$$f_{1/\delta}(i_l^{(1,k_1)}) = k_1, \quad g_{1/\delta-1}(i_l^{(1,k_1)}) = l.$$

Similarly let $\bar{i}_1^{(1,k_1)} < \bar{i}_2^{(1,k_1)} < \cdots < \bar{i}_{n_1}^{(1,k_1)}$ be all the indices in S^-_{1,k_1}, we then define

$$\bar{f}_{1/\delta}(\bar{i}_l^{(1,k_1)}) = k_1, \quad \bar{g}_{1/\delta-1}(\bar{i}_l^{(1,k_1)}) = l.$$

Each group S_{1,k_1} is then further partitioned equally into n^δ subgroups and let these groups be S_{2,k_1,k_2} for $1 \le k_2 \le n^{1/\delta}$. In general, suppose we have already derived $n^{(j-1)\delta}$ layer-$(j-1)$ groups for $2 \le j \le 1/\delta - 1$. Each layer-$(j-1)$ group, say, $S_{j-1,k_1,k_2,\cdots,k_{j-1}}$ is then further partitioned equally into n^δ subgroups. Let them be $S_{j,k_1,k_2,\cdots,k_j}$ for $1 \le k_j \le n^\delta$. It can be easily seen that each layer-j group contains $n_j = n^{1-j\delta}$ clauses. Again let $S^+_{j,k_1,k_2,\cdots,k_j}$ and $S^-_{j,k_1,k_2,\cdots,k_j}$ be the sets of indices of all the positive literals and negative literals in $S_{j,k_1,k_2,\cdots,k_j}$ respectively. Let $i_1^{(j,k_1,k_2,\cdots,k_j)} < i_2^{(j,k_1,k_2,\cdots,k_j)} < \cdots < i_{n_j}^{(j,k_1,k_2,\cdots,k_j)}$ be the indices in $S^+_{j,k_1,k_2,\cdots,k_j}$, we then define

$$f_{1/\delta-j+1}(i_l^{(j,k_1,k_2,\cdots,k_j)}) = k_j, \quad g_{1/\delta-j}(i_l^{(j,k_1,k_2,\cdots,k_j)}) = l.$$

Similarly let $\bar{i}_1^{(j,k_1,k_2,\cdots,k_j)} < \bar{i}_2^{(j,k_1,k_2,\cdots,k_j)} < \cdots < \bar{i}_{n_j}^{(j,k_1,k_2,\cdots,k_j)}$ be all the indices in $S^-_{j,k_1,k_2,\cdots,k_j}$, we then define

$$\bar{f}_{1/\delta-j+1}(\bar{i}_l^{(j,k_1,k_2,\cdots,k_j)}) = k_j, \quad \bar{g}_{1/\delta-j}(\bar{i}_l^{(j,k_1,k_2,\cdots,k_j)}) = l.$$

The above procedure stops when we derive layer-$(1/\delta - 1)$ groups with each of them containing n^δ clauses. We have the following simple observations.

Observation

1. For any $1 \le i \le n$, $1 \le f_k(i) \le n^\delta$ for $2 \le k \le 1/\delta$, and $1 \le g_k(i), \bar{g}_k(i) \le n^{k\delta}$ for $1 \le k \le 1/\delta - 1$.

2. If $(z_i \vee \neg z_h) \in C_2$, then $f_k(i) = \bar{f}_k(h)$ for $2 \le k \le 1/\delta$.

3. For any $0 \le k \le 1/\delta - 2$ and $i < i'$
   - If $f_{1/\delta}(i) = f_{1/\delta}(i')$, $f_{1/\delta-1}(i) = f_{1/\delta-2}(i')$, $\cdots$, $f_{1/\delta-k}(i) = f_{1/\delta-k}(i')$, then $g_{1/\delta-k-1}(i) < g_{1/\delta-k-1}(i')$.
   - If $\bar{f}_{1/\delta}(i) = \bar{f}_{1/\delta}(i')$, $\bar{f}_{1/\delta-1}(i) = \bar{f}_{1/\delta-2}(i')$, $\cdots$, $\bar{f}_{1/\delta-k}(i) = \bar{f}_{1/\delta-k}(i')$, then $\bar{g}_{1/\delta-k-1}(i) < \bar{g}_{1/\delta-k-1}(i')$.

4. For any $1 \le \tau \le n^\delta$ and $2 \le k \le 1/\delta$, $|\{i|f_k(i) = \tau\}| = |\{i|\bar{f}_k(i) = \tau\}| = n^{1-\delta}$.

2.2.2 Construction of the Scheduling Instance

We construct the scheduling instance based on I_{sat}. Throughout this section we set $x = 4n^\delta$, $r = 2^{3/\delta+9}n^{1+\delta}$ and use $s(j)$ to denote the processing time of job j. We will show that the constructed scheduling instance admits a feasible schedule of makespan $K = 10^5 r$ if and only if the given 3SAT' instance is satisfiable. Similar to the special case when $\delta = 1/2$, we construct $8n$ variable jobs, $8n$ truth assignment jobs, n clause jobs, $2n$ dummy jobs. The only difference is that we construct more agent jobs, indeed, we will construct $4(1/\delta - 1)n$ agent jobs, divided from layer-1 agent jobs to layer-$(1/\delta - 1)$ agent jobs.

Variable jobs: $v_{i,1}^\gamma$ and $v_{i,2}^\gamma$ are constructed for z_i, $v_{i,3}^\gamma$ and $v_{i,4}^\gamma$ are for $\neg z_i$.

$$s(v_{i,k}^T) = r + 2^{1/\delta+7}[f_{1/\delta}(i)x^{1/\delta} + i] + 2^{1/\delta+6} + k, \quad k = 1, 2$$

$$s(v_{i,k}^T) = r + 2^{1/\delta+7}[\bar{f}_{1/\delta}(i)x^{1/\delta} + i] + 2^{1/\delta+6} + k, \quad k = 3, 4$$

$$s(v_{i,k}^F) = s(v_{i,k}^T) + 2r, \quad k = 1, 2, 3, 4$$

Agent jobs: layer-j agent jobs $\eta_{i,j,+}^\gamma$ and $\eta_{i,j,-}^\gamma$ are constructed for $1 \le i \le n$ and $1 \le j \le 1/\delta - 1$.

$$s(\eta_{i,j,+}^T) = r + 2^{1/\delta+7}[\sum_{k=j+1}^{1/\delta} f_k(i)x^k + g_j(i)] + 2^{j+6} + 8, \quad j = 1, 2, \cdots, 1/\delta - 1$$

$$s(\eta_{i,j,-}^T) = r + 2^{1/\delta+7}[\sum_{k=j+1}^{1/\delta} \bar{f}_k(i)x^k + \bar{g}_j(i)] + 2^{j+6} + 16, \quad j = 2, 3, \cdots, 1/\delta - 1$$

Specifically, $s(\eta_{i,1,-}^T) = r + 2^{1/\delta+7}[\sum_{k=2}^{1/\delta} \bar{f}_k(i)x^k + \bar{g}_1(i)x] + 2^7 + 16$,

$$s(\eta_{i,j,\sigma}^F) = s(\eta_{i,\sigma}^T) + 2r, \quad \sigma = +, -$$

Truth assignment jobs: a_i^γ, b_i^γ, c_i^γ and d_i^γ.

$$s(a_i^F) = 11r + (2^7 i + 8), s(b_i^F) = 11r + (2^7 i + 32),$$

$$s(c_i^F) = 101r + (2^7 i + 16), s(d_i^F) = 101r + (2^7 i + 64),$$

$$s(\zeta_i^T) = s(\zeta_i^F) + r, \quad \zeta = a, b, c, d.$$

Clause jobs: 3 clause jobs are constructed for every $c_j \in C_1$ where $j \in R$, with one u_j^T and two copies of u_j^F:

$$s(u_j^T) = 10004r + 2^{1/\delta+9}j, s(u_j^F) = 10002r + 2^{1/\delta+9}j.$$

Dummy jobs: $n + n/3$ jobs with processing time $1000r$, and $n - n/3$ jobs with processing time $1002r$.

Let V and V_a be the set of variable jobs and agent jobs. Let A, B, C, D be the set of a_i^γ, b_i^γ, c_i^γ and d_i^γ respectively. Let $G_0 = V \cup V_a$, $G_1 = A \cup B$, $G_2 = C \cup D$, G_3 be the set of dummy jobs and $G_4 = U$ be the set of clause jobs. Again we may drop the superscript for simplicity. We construct huge jobs. They create gaps on machines. According to which jobs are needed to fill up the gap, they are divided into six groups.

Two huge jobs (variable-agent jobs) $\theta_{\eta,i,+}$ and $\theta_{\eta,i,-}$ are constructed for each variable z_i:

$$\begin{aligned}
s(\theta_{\eta,i,+}) &= 10^5 r - [4r + 2^{1/\delta+7}(2f_{1/\delta}(i)x^{1/\delta} + i + g_{1/\delta-1}(i)) + 2^{1/\delta+6} + 2^{1/\delta+5} + 10] \\
s(\theta_{\eta,i,-}) &= 10^5 r - [4r + 2^{1/\delta+7}(2\bar{f}_{1/\delta}(i)x^{1/\delta} + i + \bar{g}_{1/\delta-1}(i)) + 2^{1/\delta+6} + 2^{1/\delta+5} + 20]
\end{aligned}$$

$2/\delta - 4$ huge jobs (layer-decreasing jobs) $\theta_{i,j,+}$ and $\theta_{i,j,-}$ are constructed for $j = 1, \cdots, 1/\delta - 2$. For $j = 2, 3, \cdots, 1/\delta - 2$, their processing times are

$$\begin{aligned}
s(\theta_{i,j,+}) &= 10^5 r - [4r + 2^{1/\delta+7}(2\sum_{k=j+2}^{1/\delta} f_k(i)x^k + f_{j+1}(i)x^{j+1} + g_{j+1}(i) + g_j(i)) + 2^{j+7} + 2^{j+6} + 16] \\
s(\theta_{i,j,-}) &= 10^5 r - [4r + 2^{1/\delta+7}(2\sum_{k=j+2}^{1/\delta} \bar{f}_k(i)x^k + \bar{f}_{j-1}(i)x^{j+1} + \bar{g}_{j+1}(i) + \bar{g}_j(i)) + 2^{j+7} + 2^{j+6} + 32].
\end{aligned}$$

For $j = 1$, their processing times are

$$\begin{aligned}
s(\theta_{i,1,+}) &= 10^5 r - [4r + 2^{1/\delta+7}(2\sum_{l=3}^{1/\delta} f_l(i)x^l + f_2(i)x^2 + g_2(i) + g_1(i)) + 2^8 + 2^7 + 16] \\
s(\theta_{i,1,-}) &= 10^5 r - [4r + 2^{1/\delta+7}(2\sum_{l=3}^{1/\delta} \bar{f}_l(i)x^l + \bar{f}_2(i)x^2 + \bar{g}_2(i) + \bar{g}_1(i)x) + 2^8 + 2^7 + 32].
\end{aligned}$$

One huge job (agent-agent job) θ_{i,k,C_2} is constructed for $(z_i \vee \neg z_k) \in C_2$:

$$s(\theta_{i,k,C_2}) = 10^5 r - [4r + 2^{1/\delta+7}(2\sum_{l=2}^{1/\delta} f_l(i)x^l + \bar{g}_1(k)x + g_1(i)) + 2^8 + 24].$$

Three huge jobs (variable-clause jobs) are constructed for each $c_j \in C_1$ ($j \in R$), one for each literal: for $i = j, j+1, j+2$, if $z_i \in c_j$, we construct $\theta_{j,i,+,C_1}$, otherwise $\neg z_i \in c_j$, and we construct $\theta_{j,i,-,C_1}$.

$$s(\theta_{j,i,+,C_1}) = 10^5 r - 11005r - (2^{1/\delta+7} f_{1/\delta}(k)x^{1/\delta} + 2^{1/\delta+9} i + 2^{1/\delta+7} k + 2^{1/\delta+6} + 1),$$

$$s(\theta_{j,i,-,C_1}) = 10^5 r - 11005r - (2^{1/\delta+7} \bar{f}_{1/\delta}(k)x^{1/\delta} + 2^{1/\delta+9} i + 2^{1/\delta+7} k + 2^{1/\delta+6} + 3).$$

One huge job (variable-dummy job) is constructed for each variable. Notice that each variable appears exactly three times in clauses, if z_i appears twice while $\neg z_i$ appears once, we construct $\theta_{i,-}$. Otherwise, we construct $\theta_{i,+}$ instead.

$$s(\theta_{i,+}) = 10^5 r - 1003r - (2^{1/\delta+7} f_{1/\delta}(i)x^{1/\delta} + 2^{1/\delta+7} i + 2^{1/\delta+6} + 1),$$

$$s(\theta_{i,-}) = 10^5 r - 1003r - (2^{1/\delta+7}\bar{f}_{1/\delta}(i)x^{1/\delta} + 2^{1/\delta+7}i + 2^{1/\delta+6} + 3).$$

Thus, for each clause c_i ($i \in R$) and $k = i, i+1, i+2$, either $\theta_{i,+}$ and $\theta_{j,i,-,C_1}$ exist, or $\theta_{i,-}$ and $\theta_{j,i,+,C_1}$ exist.

Four huge jobs (variable-assignment jobs) are constructed for each variable z_i, namely $\theta_{i,a,c}$, $\theta_{i,b,d}$, $\theta_{i,a,d}$ and $\theta_{i,b,c}$:

$$s(\theta_{i,a,c}) = 10^5 r - 115r - 2^{1/\delta+7}(f_{1/\delta}(i)x^{1/\delta} + i) - (2^{1/\delta+6} + 2^8 i + 25),$$

$$s(\theta_{i,b,d}) = 10^5 r - 115r - 2^{1/\delta+7}(f_{1/\delta}(i)x^{1/\delta} + i) - (2^{1/\delta+6} + 2^8 i + 98),$$

$$s(\theta_{i,a,d}) = 10^5 r - 115r - 2^{1/\delta+7}(\bar{f}_{1/\delta}(i)x^{1/\delta} + i) - (2^{1/\delta+6} + 2^8 i + 75),$$

$$s(\theta_{i,b,c}) = 10^5 r - 115r - 2^{1/\delta+7}(\bar{f}_{1/\delta}(i)x^{1/\delta} + i) - (2^{1/\delta+6} + 2^8 i + 52).$$

The jobs we construct now are similar to that we construct in the special case, except that we construct a set of agent jobs from layer-1 to layer-$(1/\delta - 1)$ instead of only two agent jobs, and a set of layer-decreasing jobs so as to leave gaps for these agent jobs. It is easy to verify that we construct $2/\delta n + 5n$ huge jobs, and thus there are $2/\delta n + 5n$ identical machines in the scheduling instance.

The processing time of each job is a polynomial on x (except of the r-term). The reader may refer to the following tables for an overview of the coefficients

Table 3: coefficients-of-agent-jobs

Jobs/coefficients	$2^{1/\delta+7}x^{1/\delta}$	$2^{1/\delta+7}x^{1/\delta-1}$	$\cdots$	$2^{1/\delta+7}x^{j+1}$	$2^{1/\delta+7}x^{j}$	$2^{1/\delta+7}x^{j-1}$	$\cdots$	$2^{1/\delta+7}x^{2}$
$\eta_{i,j,+}$	$f_{1/\delta}(i)$	$f_{1/\delta-1}(i)$	$\cdots$	$f_{j+1}(i)$	0	0	$\cdots$	0
$\eta_{i,j-1,+}$	$f_{1/\delta}(i)$	$f_{1/\delta-1}(i)$	$\cdots$	$f_{j+1}(i)$	$f_j(i)$	0	$\cdots$	0

Table 4: coefficients-of-huge-jobs

Jobs/coefficients	$2^{1/\delta+7}x^{1/\delta}$	$2^{1/\delta+7}x^{1/\delta-1}$	$\cdots$	$2^{1/\delta+7}x^{j+1}$	$2^{1/\delta+7}x^{j}$	$2^{1/\delta+7}x^{j-1}$	$\cdots$	$2^{1/\delta+7}x^{2}$
$\theta_{i,j,+}$	$2f_{1/\delta}(i)$	$2f_{1/\delta-1}(i)$	$\cdots$	$f_{j+1}(i)$	0	0	$\cdots$	0
$\theta_{i,j-1,+}$	$2f_{1/\delta}(i)$	$2f_{1/\delta-1}(i)$	$\cdots$	$2f_{j+1}(i)$	$f_j(i)$	0	$\cdots$	0
θ_{i,k,C_2}	$2f_{1/\delta}(i)$	$2f_{1/\delta-1}(i)$	$\cdots$	$2f_{j+1}(i)$	$2f_j(i)$	$2f_{j-1}(i)$	$\cdots$	$2f_2(i)$

It is not difficult to verify that the total processing time of all the jobs is $(2/\delta n + 5n) \cdot 10^5 r$.

2.2.3 3SAT to Scheduling

We show that, if I_{sat} is satisfiable, then the makespan of the optimal solution for the constructed scheduling instance is $10^5 r$. Notice that the number of huge jobs equals the number of machines. We put one huge job on each machine. For simplicity, we may use the symbol of a huge job to denote a machine, e.g., we call a machine as machine $\theta_{i,a,c}$ if the job $\theta_{i,a,c}$ is on it.

We first schedule jobs in the following way. Recall that the superscript (T or F) of a job only influences its r-term. It is not difficult to verify that by scheduling jobs in the following way, except for the r-terms, the coefficients of other terms of jobs on the same machine add up to 0.

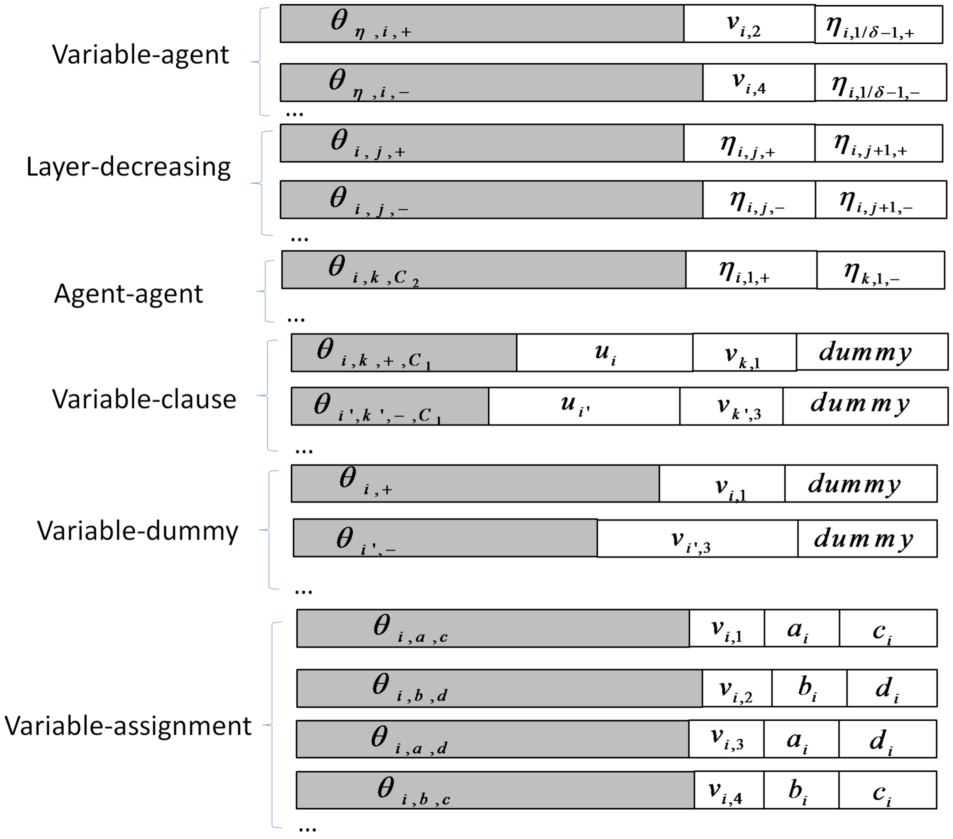

Figure 1: index-scheduling

We determine the superscripts of each job so that their r-terms add up to $10^5 r$. Suppose according to the truth assignment of I_{sat} variable z_i is true, we determine the superscripts in the following way. For ease of description, we may first assume that currently only huge jobs are scheduled, and we schedule the remaining jobs one by one as follows.

On machine $\theta_{\eta,i,+}$, we schedule $v^T_{i,2}$ and $\eta^F_{i,1/\delta-1,+}$. On machine $\theta_{i,j,+}$ where $j = 1, 2, \cdots, 1/\delta - 2$, we schedule $\eta^F_{i,j,+}$ and $\eta^T_{i,j+1,+}$. Thus, $\eta^F_{i,j,+}$ is on machine $\theta_{i,j,+}$, $\eta^T_{i,j,+}$ is on machine $\theta_{i,j-1,+}$, which means both the true job and false job of $\eta_{i,j,+}$ are scheduled. While for $\eta_{i,1,+}$, only $\eta^F_{i,1,+}$ is scheduled (on machine $\theta_{i,1,+}$).

Similarly on machine $\theta_{\eta,i,-}$, we schedule $v^F_{i,4}$ and $\eta^T_{i,1/\delta-1,-}$. On machine $\theta_{i,j,-}$ where $j = 1, 2, \cdots, 1/\delta - 2$, we schedule $\eta^T_{i,j,-}$ and $\eta^F_{i,j+1,-}$. Thus, $\eta^T_{i,j,-}$ is on machine $\theta_{i,j,-}$, $\eta^F_{i,j,-}$ is on machine $\theta_{i,j-1,-}$, which means both the true job and false job is scheduled. While for $\eta_{i,1,-}$, only $\eta^T_{i,1,-}$ is scheduled (on machine $\theta_{i,1,-}$).

Consider agent-agent machines. For the variable z_i, there is a clause $(z_i \vee \neg z_k) \in C_2$ for some k, and we need to schedule $\eta_{i,1,+}$ and $\eta_{k,1,-}$ on machine θ_{i,k,C_2}. Since I_{sat} is satisfiable, variables z_i and z_k should be both true or both false. Thus, given that z_i is true, z_k is also true. This implies that $\eta^T_{i,1,+}$ and $\eta^F_{k,1,-}$ are not scheduled before, and we schedule these two jobs.

Meanwhile in C_2 there is also a clause $(z_{k'} \vee \neg z_i)$ for some k', and we need to schedule $\eta_{i,1,-}$ and $\eta_{k',1,+}$ on machine θ_{k',i,C_2}. Since I_{sat} is satisfiable, variables $z_{k'}$ and z_i should be both true or both false. Thus, given that z_i is true, z_k is also true. This implies that $\eta^T_{k',1,+}$ and $\eta^F_{i,1,-}$ are not unscheduled before, and we schedule them on machine θ_{k',i,C_2}.

Consider variable-assignment jobs. We put $v^F_{i,1}$, a^F_i, c^F_i on machine $\theta_{i,a,c}$, put $v^F_{i,2}$, b^F_i, d^F_i on machine $\theta_{i,b,d}$, put $v^T_{i,3}$, a^T_i, d^T_i on machine $\theta_{i,a,d}$, and put $v^T_{i,4}$, b^T_i, c^T_i on machine $\theta_{i,b,c}$. Thus, both the true copy and false copy of a_i, b_i, c_i and d_i are scheduled. It can be easily seen that the r-terms of three true jobs or three false jobs both add up to $115r$. Otherwise, z_i is false, and we schedule jobs just in the opposite way, i.e., we replace each true job with its corresponding false job, and each false job with its corresponding true job in the previous scheduling.

We consider the remaining jobs. If z_i is true, then $v^T_{i,1}$ and $v^F_{i,3}$ are left. If z_i is false, then $v^F_{i,1}$ and $v^T_{i,3}$ are left. These jobs should be scheduled with clause jobs and dummy jobs on variable-clause machines or variable-dummy machines. Notice that for any $i \in R$ and $k \in \{i, i+1, i+2\}$, either $\theta_{i,k,+,C_1}$ and $\theta_{k,-}$ exist, or $\theta_{i,k,-,C_1}$ and $\theta_{k,+}$ exist.

Suppose the variable z_k is true. If $\theta_{i,k,+,C_1}$ and $\theta_{k,-}$ exist, we put $v^T_{k,1}$ on machine $\theta_{i,k,+,C_1}$, and $v^F_{k,3}$ on machine $\theta_{k,-}$. Otherwise $\theta_{i,k,-,C_1}$ and $\theta_{k,+}$ exist, and we put $v^F_{k,3}$ on machine $\theta_{i,k,-,C_1}$, and $v^T_{k,1}$ on machine $\theta_{k,+}$. In both cases, the remaining jobs $v^T_{k,1}$ and $v^F_{k,3}$ are scheduled.

Otherwise z_k is false. If $\theta_{i,k,+,C_1}$ and $\theta_{k,-}$ exist, put $v^F_{k,1}$ on machine $\theta_{i,k,+,C_1}$, and $v^T_{k,3}$ on machine $\theta_{k,-}$. Otherwise $\theta_{i,k,-,C_1}$ and $\theta_{k,+}$ exist. We put $v^T_{k,3}$ on machine $\theta_{i,k,-,C_1}$, and $v^F_{k,1}$ on machine $\theta_{k,+}$. Again in both cases, the remaining jobs $v^F_{k,1}$ and $v^T_{k,3}$ are scheduled.

From now on we drop the symbol $+$ or $-$ and just use θ_{i,k,C_1} to denote either $\theta_{i,k,+,C_1}$ or $\theta_{i,k,-,C_1}$, and use θ_k to denote either $\theta_{k,+}$ or $\theta_{k,-}$. It is easy to verify that the above scheduling has the following property.

Property If c_i is satisfied by variable z_k (i.e., $z_k \in c_i$ and z_k is true or $\neg z_k \in c_i$ and z_k is false), then a true variable job is on machine θ_{i,k,C_1}; if c_i is not satisfied by z_k, then a false variable job is on machine θ_{i,k,C_1}.

Consider variable-dummy machines. For each $k = 1, 2, \cdots, n$, there is one machine θ_k. If a true variable job is on it, we then put additionally a dummy job of size $1002r$. Otherwise a false variable job is on it, and we put additionally a dummy job of size $1000r$ on it. Thus, in both cases the r-terms of variable job and dummy job add up to $1003r$.

Consider variable-clause machines. For each clause $c_i \in C_1$ (i.e., $i \in R$), there are three copies of u_i, one true and two false. There are three machines, θ_{i,i,C_1}, $\theta_{i,i+1,C_1}$ and $\theta_{i,i+2,C_1}$.

Notice that according to the truth assignment, c_i is satisfied by at least one variable. Suppose c_i is satisfied by z_{k_1}, and let z_{k_2} and z_{k_3} be the remaining two variables in this clause, i.e., k_1, k_2, k_3 is some permutation of the three indices $i, i+1, i+2$. We put u_i^T on machine θ_{i,k_1,C_1}. Additionally, we put a dummy job of size $1000r$ on this machine. According to the property we have mentioned above, since c_i is satisfied by z_{k_1}, the variable job on machine θ_{i,k_1,C_1} is a true job. Thus, the r-terms of the true clause job, true variable job and a dummy job on θ_{i,k_1,C_1} add up to $11005r$.

Consider machine θ_{i,k_2,C_1} and θ_{i,k_3,C_1}. We put one of the remaining two false jobs u_i^F on them respectively. We add dummy jobs according to the following criteria. If the variable job is true, we add a dummy job of size $1002r$. If the variable job is false, we add a dummy job of size $1000r$.

Thus in both cases, the r-terms of the variable job and dummy job add up to $1003r$. And if we further add the r-terms of the false clause job and the relation job, the sum is $10^5 r$. Finally we check the number of dummy jobs that are used.

For simplicity we use $(T/F, T/F, 1000r/1002r)$ to denote the truth-type of a variable-clause machine, i.e., the first coordinate is T is the variable job is true, and F if it is false, similarly the second coordinate is T (or F) if the clause job is T (or F), the third coordinate is $1000r$ (or $1002r$) if the dummy job is of size $1000r$ (or $1002r$). We also denote the truth-type of a variable-dummy machine in the form of $(T/F, 1000r/1002r)$.

A dummy job of size $1000r$ is always scheduled on a machine of truth-type $(T, T, 1000r)$, $(F, F, 1000r)$ and $(F, 1000r)$, while a dummy job of $1002r$ is scheduled on a machine of truth-type $(T, F, 1002r)$ and $(T, 1002r)$. Notice that on these machines, there are n true variable jobs and n false variable jobs, and there are $|C_1| = n/3$ true clause jobs, thus simple calculations show that $n + n/3$ dummy jobs of $1000r$ and $n - n/3$ dummy jobs of $1002r$ are scheduled, which coincides with the dummy jobs we construct.

2.2.4 Scheduling to 3SAT

We show that, if the constructed scheduling instance admits a feasible schedule with makespan $10^5 r$, then I_{sat} is satisfiable. Notice that in a scheduling problem, jobs are represented by their processing times rather than symbols, we first show that we can the processing time of each job we construct is distinct (except that two copies of u_i^F are constructed for every clause in C_1), this would be enough to determine the symbol of a job from its processing time.

Distinguishing Jobs from Their Processing Times Recall that we define the processing time of a job in the form of a polynomial, we use the notion x^j-term or r-term in their direct meaning. Meanwhile, we call the sum of all except the r-term of a job as the small-r-term. For any $2 \leq j \leq 1/\delta$, we delete the r-term and x^k-term with

$k \geq j$ from the processing time of a job, and call the sum of all the remaining terms as the small-x^j-term.

For example, the relation job $\theta_{i,3,+}$ is of processing time $10^5r - [4r + 2^{1/\delta+7}(2\sum_{k=5}^{1/\delta} f_k(i)x^k + f_4(i)x^4 + g_4(i) + g_3(i)) + 2^{10} + 2^9 + 16]$, and thus for $5 \leq k \leq 1/\delta$, its x^k-term is $2^{1/\delta+7} \cdot 2f(i)x^k$. Its x^4-term is $f(4)(i)x^4$. Its x^3-term and x^2-term are 0. Its small-x^5-term is $2^{1/\delta+7}(f_4(i)x^4 + g_4(i) + g_3(i)) + 2^{10} + 2^9 + 16$. Meanwhile, for a clause job, say, u_i, its x^j-term is 0 for $1 \leq j \leq 1/\delta$, and its small-x^j-term for any j is $2^{1/\delta+9}i$.

Consider the small-r-term of any job. If it is a huge job, this value is negative and its absolute value is bounded by $2^{1/\delta+7}(2\sum_{k=2}^{1/\delta} n^{1/\delta}x^k + 2n) + 2^{1/\delta+7} + 32 < 1/2r$ (notice that $n^\delta x^k = 1/4x^{k+1}$). Otherwise it is a variable, or agent, or clause, or truth assignment, or dummy job, and the sum is positive with its absolute value also bounded by $2^{1/\delta+7}(\sum_{k=2}^{1/\delta} n^{1/\delta}x^k + n) + 2^{1/\delta+6} + 64 < 1/4r$.

For the small-x^j-terms of jobs, we have the following lemma.

Lemma 8 *For a huge job, its small-x^j-term ($2 \leq j \leq 1/\delta$) is negative, and the absolute value is bounded by $2^{1/\delta+7} \cdot 3/4x^j$. For a variable or agent job, its small-x^j-term is positive and bounded by $2^{1/\delta+7} \cdot 3/8x^j$.*

Proof. Notice that $g_j(i) \leq n^{j\delta}$, while $f_j(i)x^j \geq 2^{2j}n^{j\delta} > g_j(i)$ for any $2 \leq j \leq 1/\delta - 1$. Thus for a huge job, its small-x^j-term is at most

$$\begin{array}{rl}
 & 2^{1/\delta+7}[2\sum_{l=2}^{j-1} f_l(i)x^l + \bar{g}_1(k)x + g_1(i)] + 2^{1/\delta+6} + 2^{1/\delta+5} + 32 \\
\leq & 2^{1/\delta+7}[2\sum_{l=2}^{j-1} n^\delta x^l + n^\delta x + n^\delta + 1] \\
\leq & 2^{1/\delta+7}[2\sum_{l=2}^{j-1} n^\delta x^l + 2n^\delta x] \\
\leq & 2^{1/\delta+7}[2\sum_{l=3}^{j-1} n^\delta x^l + 3n^\delta x^2] \\
\leq & 2^{1/\delta+7}[2\sum_{l=4}^{j-1} n^\delta x^l + 3n^\delta x^3] \\
\cdots & \\
\leq & 2^{1/\delta+7} \cdot 3n^\delta x^{j-1} \\
\leq & 2^{1/\delta+7} \cdot 3/4x^j
\end{array}$$

The inequalities make use of the simple observation that $n^\delta x^k = 1/4x^{k+1}$ for $1 \leq k \leq 1/\delta$. The proof for variable or agent jobs is similar. □

Given the processing time of a job, we can easily determine whether it is a huge, variable, agent, clause, or dummy job by considering its quotient of divided by r, and the residual of divided by 2^7, and if it is a huge job, we may further determine if it is a variable-agent, layer-decreasing, agent-agent, variable-clause, variable-dummy, variable-assignment job. Using the above lemma, if it is a variable, or agent, or clause, or dummy job, we can easily expand it into the summation form and determine its symbol according to Observation 3.

Suppose we are given the processing time of a huge jobs. Again it is easy to determine its symbol if it is a variable-assignment, variable-dummy or variable-clause job. If it is an agent-agent job, then according to the fact that $g_1(i) \leq n^{\delta} \leq 1/4x$, we can also expand the processing time into the summation form and determine its symbol. If it is a variable-agent or layer-decreasing job, we show that the processing time of such a job is unique.

Suppose $s(\theta_{i_1,j_1,+}) = s(\theta_{i_2,j_2,+})$, then according to Lemma 8 we have $j_1 = j_2 = j$ and $f_k(i_1) = f_k(i_2)$ for $j+1 \leq k \leq 1/\delta$ and $g_{j+1}(i_1) + g_j(i_1) = g_{j+1}(i_2) + g_j(i_2)$. Now according to Observation 3, we have $i_1 = i_2$. Similarly if $s(\theta_{i_1,j_1,-}) = s(\theta_{i_2,j_2,-})$, we can also prove that $i_1 = i_2$, $j_1 = j_2$. Obviously it is impossible that $s(\theta_{i_1,j_1,+}) = s(\theta_{i_2,j_2,-})$. The proof for variable-agent jobs is similar.

The following part of this subsection is devoted to proving the following lemma.

Lemma 9 *If there is a solution for the constructed scheduling instance in which the load of each machine is $10^5 r$, then I_{sat} is satisfiable.*

Let Sol^* be an optimal solution, it can be easily seen that there is a huge job on each machine, leaving a gap if the load of each machine is $10^5 r$. We may use the symbol of a huge job to denote the corresponding gap and the machine it is scheduled on.

We divide jobs into groups based on their processing times. According to the previous subsection, we know the processing time of a variable or agent job is either in $[r, 5/4r]$ or in $[3r, 13/4r]$. Let G_0 be the set of them. The processing time of a_i or b_i belongs to $[11r, 12.5r]$, of c_i or d_i belongs to $[101r, 102.5r]$. Let $G_1 = A \cup B$, $G_2 = C \cup D$.

Lemma 10 *In Sol^*, besides the huge job, the other jobs on a machine are:*

- *The variable-agent, or layer-decreasing, or agent-agent gap is filled up by two jobs of G_0.*
- *The variable-clause gap is filled up by one clause job, one dummy job and one job of G_0.*
- *The variable-dummy gap is filled up by one dummy job and one job of G_0.*
- *The variable-assignment gap is filled up by one job of $G_1 = A \cup B$, one job of $G_2 = C \cup D$, and one job of G_0.*

Proof. See the following table (Table 5) as an overview of gaps on machines (here Θ_0 denotes the set of variable-agent, layer-decreasing and agent-agent gaps).

Consider clause jobs. According to the table they can only be used to fill variable-clause gaps. Meanwhile each variable-clause machine (gap) could accept at most one clause job. Notice that there are n clause jobs and n variable-clause machines, thus there is one clause job on every variable-clause machine. By further subtracting the processing time of the clause job from the gap, the remaining gap of a variable-clause machine belongs to $[1000r, 1004r]$.

Consider dummy jobs. According to the current gaps, they can only be scheduled on variable-clause or variable-dummy machines, and each of these machines could accept at most one dummy job. Again notice that there are $2n$ such machines and $2n$ dummy jobs, there is one dummy job on every variable-clause and variable-dummy machine. The current gap of a variable-clause machine is in $[0, 4r]$, of a variable-dummy machine is in $[r, 4r]$. Using the same argument we can show that there is one job of $C \cup D$ and one job of $A \cup B$ on each variable-assignment machine.

Consider variable and agent jobs. Each machine of Θ_0 has a gap in $(4r, 5r)$, implying that there are at least two variable or agent jobs on it. The current gap of a variable-assignment machine is at least $115r-(102r+2^7n+12r+2^7n+64+64) \geq r-2^9n > 1/2r$, thus there is at least one variable or agent job on it. Similarly there is at least one variable or agent job on a variable-dummy machine.

Consider each variable-clause machine. As we have determined, there are a clause and a dummy job on it. We check their total processing times more carefully. By subtracting the huge job in from 10^5r, the gap is in $[11005r, (11005+1/2)r]$. If the clause job on this machine is a true job, with a processing time over $10004r$, then the dummy job on it can only be of $1000r$, otherwise the total processing time of the two jobs is over $11006r$, which is a contradiction. Thus, the total processing time of the two jobs is at most $11004r + 2^{1/\delta+9}n + 1000r \leq (11004 + 1/2)r$, which means there is at least one variable or agent job on this machine. Otherwise, the clause job on this machine is a false job with a processing time at most $10002r + 2^{1/\delta+9}n \leq (10002 + 1/2)r$. Adding a dummy job, their total processing time is at most $(11004 + 1/2)r$, and again we can see that there is at least one variable or agent job on this machine.

The above analysis shows that there is at least one job of G_0 on a variable-clause, variable-dummy and variable-assignment machine, and at least two jobs of G_0 on each machine of Θ_0, requiring $4n + 4/\delta n$ jobs, which equals to $|G_0|$. Thus the lemma follows directly. □

Given the above lemma, we consider the residuals of each job divided by $2^{1/\delta+7}$. The fact that the three or four residuals on each machine should add up to 0 implies the following table (Table 6).

Table 5: Sizes of gaps

Machines(Gaps)	Θ_0	Variable-clause	Variable-dummy	Variable-assignment
Size of Gaps	$(4r, 5r)$	$(11005r, 11006r)$	$(1003r, 1004r)$	$(115r, 116r)$

The next step is to characterize the indices, i.e., we need to prove that for each row, $i = i' = i''$ (or $i = i_1 = i_2 = i_3$). If the indices equal for jobs on a machine, this machine (gap) is called satisfied. The above table, combined with Lemma 8, implies the following lemma.

Lemma 11 *For jobs on each machine, their r-terms add up to $10^5 r$, x^k-terms ($2 \le k \le 1/\delta - 1$) add up to 0.*

The x^k-term of each huge job is negative and should be canceled by the corresponding terms from other jobs. Similar as the proof for the special case when $\delta = 1/2$, we would divide the x^k-terms ($2 \le k \le 1/\delta$) of each huge job (gap) into singular terms and regular terms. Notice that here we use the notion of singular (regular) terms instead of singular (regular) gaps because when $1/\delta > 2$ we need to consider multiple terms of a gap.

We define singular (regular) terms in the following way. The $x^{1/\delta}$-terms of variable-clause, variable-dummy and variable-assignment gaps are singular terms.

For other gaps, see Table 7. The terms marked with $*$ are singular term (e.g., the x^j-term of $\theta_{i,j-1,\sigma}$), all the other terms are regular terms.

A singular term of a gap, say, $2^{1/\delta+7}\tau x^j$ for $1 \le \tau \le n^\delta$, is called well-canceled, if it is filled up by one job with the x^j-term of $2^{1/\delta+7}\tau x^j$ and other jobs with the x^j-terms of 0. A regular term, say, $2^{1/\delta+7} \cdot 2\tau x^j$ for $1 \le \tau \le n^\delta$, is called well-canceled, if it is filled up by two jobs whose x^j-terms are $2^{1/\delta+7}\tau x^j$.

Lemma 12 *Every singular term is well-canceled.*

The proof is straightforward.

Lemma 13 *Every regular term is well-canceled.*

Before we prove this lemma, we first count the number of variable and agent jobs whose x^k-term is $2^{1/\delta+7} \cdot \tau_k x^k$ where $2 \le k \le 1/\delta$ and $1 \le \tau_k \le n^\delta$. For simplicity we call them as τ_k-jobs. According to Observation 4, $|\{i|f_k(i) = \tau_k\}| = |\{i|\bar{f}_k(i) = \tau_k\}| = n^{1-\delta} = n_1$, thus we have Table 8.

The factor 2 in the last row comes from the fact that for each symbol there are actually a true job and a false job, and thus the numbers should double. We call the gap whose x^k-term is a regular term and equals to $2^{1/\delta+7} \cdot 2\tau_k x^k$ as a regular τ_k-gaps, and call the gap whose x^k-term is a singular term and equals to $2^{1/\delta+7} \cdot \tau_k x^k$ as a singular τ_k-gap. We count their numbers. See Table 9 as an overview.

Notice that in Table 9 we do not list variable-clause, variable-dummy and variable-assignment gaps, however, they contribute to the number of singular $2^{1/\delta+7}\tau_{1/\delta}x^{1/\delta}$ terms by $6n_1$ for any $1 \le \tau_{1/\delta} \le n^\delta$. Now we come to the proof of Lemma 13.

Proof. We prove the lemma through induction. We first consider $x^{1/\delta}$-terms. A regular $x^{1/\delta}$-term of a gap could always be expressed as $2^{1/\delta+7} \cdot 2\tau_{1/\delta}x^{1/\delta}$ for $1 \le \tau_{1/\delta} \le n^\delta$.

Table 6: Overview of jobs on machines

$\theta_{\eta,i,+}$	$v_{i',2}$	$\eta_{i'',1/\delta-1,+}$	$\backslash$
$\theta_{\eta,i,-}$	$v_{i',4}$	$\eta_{i'',1/\delta-1,-}$	$\backslash$
$\theta_{i,j,+}$	$\eta_{i',j,+}$	$\eta_{i'',j+1,+}$	$\backslash$
$\theta_{i,j,-}$	$\eta_{i',j,-}$	$\eta_{i'',j+1,-}$	$\backslash$
θ_{i,k,C_2}	$\eta_{i',1,+}$	$\eta_{i'',1,-}$	$\backslash$
$\theta_{i,k,+,C_1}$	$u_{i'}$	$v_{i'',1}$	dummy
$\theta_{i,k,-,C_1}$	$u_{i'}$	$v_{i'',3}$	dummy
$\theta_{i,+}$	$v_{i',1}$	dummy	$\backslash$
$\theta_{i,-}$	$v_{i',3}$	dummy	$\backslash$
$\theta_{i,a,c}$	$v_{i_1,1}$	a_{i_2}	c_{i_3}
$\theta_{i,b,d}$	$v_{i_1,2}$	b_{i_2}	d_{i_3}
$\theta_{i,a,d}$	$v_{i_1,3}$	a_{i_2}	d_{i_3}
$\theta_{i,b,c}$	$v_{i_1,4}$	a_{i_2}	d_{i_3}

Table 7: Singular and regular terms

Gaps/Coefficients	$2^{1/\delta+7}x^{1/\delta}$	$2^{1/\delta+7}x^{1/\delta-1}$	$\cdots$	$2^{1/\delta+7}x^{j+1}$	$2^{1/\delta+7}x^{j}$	$2^{1/\delta+7}x^{j-1}$	$\cdots$	$2^{1/\delta+7}x^{2}$
$\theta_{\eta,i,+}$	$2f_{1/\delta}(i)$	0	$\cdots$	0	0	0	$\cdots$	0
$\theta_{\eta,i,-}$	$2\bar{f}_{1/\delta}(i)$	0	$\cdots$	0	0	0	$\cdots$	0
$\theta_{i,j-1,+}$	$2f_{1/\delta}(i)$	$2f_{1/\delta-1}(i)$	$\cdots$	$2f_{j+1}(i)$	$f_j(i)^*$	0	$\cdots$	0
$\theta_{i,j-1,-}$	$2\bar{f}_{1/\delta}(i)$	$2\bar{f}_{1/\delta-1}(i)$	$\cdots$	$2\bar{f}_{j+1}(i)$	$\bar{f}_j(i)^*$	0	$\cdots$	0
θ_{i,k,C_2}	$2f_{1/\delta}(i)$	$2f_{1/\delta-1}(i)$	$\cdots$	$2f_{j+1}(i)$	$2f_j(i)$	$2f_{j-1}(i)$	$\cdots$	$2f_2(i)$

Table 8: Counting numbers of variable and agent jobs

Jobs/Coefficients	$2^{1/\delta+7}x^{1/\delta}$	$2^{1/\delta+7}x^{1/\delta-1}$	$\cdots$	$2^{1/\delta+7}x^{j}$	$\cdots$	$2^{1/\delta+7}x^{2}$
$v_{i,\iota}(\iota=1,2,3,4)$	$f_{1/\delta}(i), \bar{f}_{1/\delta}(i)$	0	$\cdots$	0	$\cdots$	0
$\eta_{i,1/\delta-1,+}, \eta_{i,1/\delta-1,-}$	$f_{1/\delta}(i), \bar{f}_{1/\delta}(i)$	0	$\cdots$	0	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\eta_{i,j-1,+}, \eta_{i,j-1,-}$	$f_{1/\delta}(i), \bar{f}_{1/\delta}(i)$	$f_{1/\delta-1}(i), \bar{f}_{1/\delta-1}(i)$	$\cdots$	$f_j(i), \bar{f}_j(i)$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\eta_{i,1,+}, \eta_{i,1,-}$	$f_{1/\delta}(i), \bar{f}_{1/\delta}(i)$	$f_{1/\delta-1}(i), \bar{f}_{1/\delta-1}(i)$	$\cdots$	$f_j(i), \bar{f}_j(i)$	$\cdots$	$f_2(i), \bar{f}_2(i)$
$\sharp$ τ_k-jobs	$2(2n_1/\delta+2n_1)$	$2\times 2(1/\delta-2)n_1$	$\cdots$	$2\times 2(j-1)n_1$	$\cdots$	$2\times 2n_1$

We start with $\tau_{1/\delta} = 1$. Notice that a regular $x^{1/\delta}$-term comes from a variable-agent, layer-decreasing or agent-agent gap. According to Table 5, the $x^{1/\delta}$-term of the other two jobs (variable or agent jobs) used to fill up such a gap are nonzero and at least $2^{1/\delta+7}x^{1/\delta}$, thus the regular term $2^{1/\delta+7} \cdot 2x^{1/\delta}$ is well canceled. Suppose for any $\tau_{1/\delta} < h_0 \le n^\delta$, each regular term $2^{1/\delta+7}\tau_{1/\delta}x^{1/\delta}$ is well-canceled.

We consider the case that $\tau_{1/\delta} = h_0$. For any $\tau_{1/\delta}$ such that $1 \le \tau_{1/\delta} < h_0$, there are in all $4n_1(1/\delta + 1)$ variable or agent jobs whose $x^{1/\delta}$-term is $2^{1/\delta+7}\tau_{1/\delta}x^{1/\delta}$ (see Table 8). We determine the scheduling of these jobs.

Among them $6n_1$ jobs are on used to cancel singular terms according to Lemma 12. Meanwhile since there are $2n_1/\delta - n_1$ gaps with regular terms $2^{1/\delta+7} \cdot 2\tau_{1/\delta}x^{1/\delta}$ (see Table 9), the induction hypothesis implies that $4n_1/\delta - 2n_1$ of these variable and agent jobs are used to cancel these regular terms.

Thus, we can conclude that for a regular $x^{1/\delta}$-term being $2^{1/\delta+7} \cdot 2h_0x^{1/\delta}$, both of the $x^{1/\delta}$ term of the two jobs (variable or agent jobs) used to cancel it are at least $2^{1/\delta+7} \cdot h_0x^{1/\delta}$. This implies, again, that the regular term $2^{1/\delta+7} \cdot 2h_0x^{1/\delta}$ is well-canceled. The proof for regular x^k-terms are the same. □

Next we prove that in *Sol**, every machine is satisfied. See Figure 2.2.3 as an illustration of such a solution. Obviously a variable-dummy machine (gap) is satisfied.

Lemma 14 *Agent-agent machines (gaps) are satisfied.*

Proof. Consider each agent-agent machine, say, θ_{i_0,k_0,C_2}. We can assume that the other two jobs on it are $\eta_{i,1,+}$ and $\eta_{k,1,-}$. Then according to Lemma 13, we have

$$f_l(i) = \bar{f}_l(k) = f_l(i_0) = \bar{f}_l(k_0), \quad l = 2, 3, \cdots, 1/\delta$$
$$g_1(i) + \bar{g}_1(k)x = g_1(i_0) + \bar{g}_1(k_0)x.$$

Since $x = 4n^\delta$, while $g_1(i), g_1(i_0), \bar{g}_1(k), \bar{g}_1(k_0) \le n^\delta$, thus $g_1(i) = g_1(i_0)$, $\bar{g}_1(k) = \bar{g}_1(k_0)$. According to the construction of functions f and g (see Observation 3), we know that $i = i_0$ and $k = k_0$. □

Table 9: Counting the number of gaps

Gaps/Coefficients	$2^{1/\delta+7}x^{1/\delta}$	$2^{1/\delta+7}x^{1/\delta-1}$	$\cdots$	$2^{1/\delta+7}x^j$	$\cdots$	$2^{1/\delta+7}x^2$
$\theta_{i,1/\delta-1,+}, \theta_{i,1/\delta-1,-}$	$2f_{1/\delta}(i), 2\bar{f}_{1/\delta}(i)$	0	$\cdots$	0	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\theta_{i,j-1,+}, \theta_{i,j-1,-}$	$2f_{1/\delta}(i), 2\bar{f}_{1/\delta}(i)$	$2f_{1/\delta-1}(i), 2\bar{f}_{1/\delta-1}(i)$	$\cdots$	$f_j(i), \bar{f}_j(i)$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\theta_{i,1,+}, \theta_{i,1,-}$	$2f_{1/\delta}(i), 2\bar{f}_{1/\delta}(i)$	$2f_{1/\delta-1}(i), 2\bar{f}_{1/\delta-1}(i)$	$\cdots$	$2f_j(i), 2\bar{f}_j(i)$	$\cdots$	$f_2(i), \bar{f}_2(i)$
$\sharp$ singular τ_k-gaps	$6n_1$	$2n_1$	$\cdots$	$2n_1$	$\cdots$	$2n_1$
$\sharp$ regular τ_k-gaps	$2n_1/\delta - n_1$	$2n_1(1/\delta - 1) - 3n_1$	$\cdots$	$2jn_1 - 3n_1$	$\cdots$	n_1

We consider variable-clause machines. Notice that for each i_0 and $k_0 \in \{i_0, i_0+1, i_0+2\}$, either $\theta_{i_0,k_0,+,C_1}$ or $\theta_{i_0,k_0,-,C_1}$ exists.

Lemma 15 *Machine $\theta_{1,k,+,C_1}$ or $\theta_{1,k,-,C_1}$ ($k = 1, 2, 3$) is satisfied. The machine $\theta_{i_0,k_0,+,C_1}$ or $\theta_{i_0,k_0,-,C_1}$ for $i_0 \geq 2$ and $k_0 \in \{i_0, i_0 + 1, i_0 + 2\}$ is satisfied if:*

- *For $i < i_0$, each machine $\theta_{i,k,+,C_1}$ or $\theta_{i,k,-,C_1}$ is satisfied.*
- *All variable jobs $v_{k',\iota}$ with $k' < i_0$ and $\iota = 1, 2, 3, 4$ are not scheduled on this machine.*

Proof. We consider clause $c_1 \in C_1$. As c_1 contains three variables z_1, z_2 and z_3, there are three huge jobs $\theta_{1,1,\sigma_1,C_1}$, $\theta_{1,2,\sigma_2,C_1}$ and $\theta_{1,3,\sigma_3,C_1}$ where $\sigma_1, \sigma_2, \sigma_3 \in \{+, -\}$. Meanwhile there are three clause jobs of u_1.

For $i_0 = 1$ and any $k_0 \in \{1, 2, 3\}$, suppose $\theta_{1,k_0,+,C_1}$ exists, and the two jobs together with it are a clause job u_i and a variable job $v_{k,\iota}$ with $\iota \in \{1, 2, 3, 4\}$. Since $s(\theta_{1,k_0,+,C_1}) = 10^5 r - 11005r - (2^{1/\delta+7} f_{1/\delta}(1) + 2^{1/\delta+9} + 2^{1/\delta+7} k_0 + 2^{1/\delta+6} + 1)$, according to Lemma 11, we have $2^{1/\delta+9} i + 2^{1/\delta+7} k + 1 = 2^{1/\delta+9} + 2^{1/\delta+7} k_0 + \iota$. If $i \geq 2$, then the left side is at least $2^{1/\delta+10}$, while the right side is at most $2^{1/\delta+9} + 2^{1/\delta+7} \times 3 + 4 < 2^{1/\delta+10}$, which is a contradiction. Thus $i = 1$ and it follows directly that $k = k_0$, $\iota = 1$. Otherwise $\theta_{1,k_0,-,C_1}$ exists, and the proof is just similar. Thus, machine $\theta_{1,k_0,+,C_1}$ or $\theta_{1,k_0,-,C_1}$ ($k_0 = 1, 2, 3$) is satisfied.

When $i_0 \geq 2$ and $k_0 \in \{i_0 + 1, i_0 + 2, i_0 + 3\}$, again we suppose that $\theta_{i_0,k_0,+,C_1}$ exists. Notice that for any $i \leq i_0 - 1$, c_i contains three variables. According to the hypothesis, the three clause jobs u_i are scheduled on three machines, they are $\theta_{i,i,+,C_1}$ or $\theta_{i,i,-,C_1}$, $\theta_{i,i+1,+,C_1}$ or $\theta_{i,i+1,-,C_1}$ and $\theta_{i,i+2,+,C_1}$ or $\theta_{i,i+2,-,C_1}$. Thus when we consider machine $\theta_{i_0,k_0,+,C_1}$, all clause jobs u_i with $i \leq i_0 - 1$ could not be scheduled on this machine.

Again suppose that the two jobs scheduled together with $\theta_{i_0,k_0,+,C_1}$ are $u_{i'}$ and $v_{k',\iota}$, then $2^{1/\delta+9} i_0 + 2^{1/\delta+7} k_0 + 1 = 2^{1/\delta+9} i' + 2^{1/\delta+7} k' + \iota$. Since $i' \geq i_0 - 1$ and $k' \geq i'$, if $i' \geq i_0 + 1$, then we have $2^{1/\delta+9} i' + 2^{1/\delta+7} k' + \sigma > 2^{1/\delta+9}(i_0 + 1) + 2^{1/\delta+7}(i_0 + 1) \geq 2^{1/\delta+9} i_0 + 2^{1/\delta+7}(i_0 + 3) + 1$, which is a contradiction. Thus $i' = i_0$, $k' = k_0$ and $\iota = 1$, which means machine $\theta_{i_0,k_0,+,C_1}$ is satisfied.

Similarly if $\theta_{i_0,k_0,-,C_1}$ exists, this machine is also satisfied. □

Lemma 16 *Machines $\theta_{1,a,c}$, $\theta_{1,b,d}$, $\theta_{1,a,d}$ and $\theta_{1,b,c}$ are satisfied. Moreover, machines $\theta_{i_0,a,c}$, $\theta_{i_0,b,d}$, $\theta_{i_0,a,d}$ and $\theta_{i_0,b,c}$ for $i_0 \geq 2$ are satisfied if:*

- *Machines $\theta_{i,a,c}$, $\theta_{i,b,d}$, $\theta_{i,a,d}$ and $\theta_{i,b,c}$ are satisfied for $i \leq i_0 - 1$.*
- *All variable jobs $v_{i',\iota}$ with $i' < i_0$ and $\iota \in \{1, 2, 3, 4\}$ are not scheduled on these machines.*

Proof. Consider machine $\theta_{1,a,c}$. Except the huge job, let the other three jobs be $v_{i_1,\iota}$ ($\iota \in \{1,2,3,4\}$), a_{i_2} and c_{i_3}. Then we have

$$2^{1/\delta+7}i_1 + 2^{1/\delta+6} + \iota_1 + 2^7 i_2 + 8 + 2^7 i_3 + 16 = 2^{1/\delta+7} + 2^{1/\delta+6} + 2^8 + 25.$$

It can be easily seen that $i_1 = i_2 = i_3 = 1$ and $\iota = 1$. Thus, machine $\theta_{1,a,c}$ is satisfied. Using similar arguments we can show that machines $\theta_{1,b,c}$, $\theta_{1,a,d}$ and $\theta_{1,b,d}$ are satisfied.

The proof that machines $\theta_{i_0,a,c}$, $\theta_{i_0,b,d}$, $\theta_{i_0,a,d}$ and $\theta_{i_0,b,c}$ are satisfied for $i_0 \geq 2$ if two conditions of the lemma hold is the same. □

For simplicity, we call variable jobs v_{i,ι_1} with $\iota_1 \in \{1,2,3,4\}$ and agent jobs η_{i,j,ι_2} with $\iota_2 \in \{+,-\}$ as jobs of index-level i.

In contrast, let $\sigma \in \{+,-\}$, we call machine $\theta_{\eta,i,\sigma}$, $\theta_{i,j,\sigma}$, machine θ_{i',i,σ,C_1}, machine $\theta_{i,\sigma}$, machines $\theta_{i,a,c}$, $\theta_{i,a,d}$, $\theta_{i,b,c}$, $\theta_{i,b,d}$ as machines of index-level i.

Specifically, machine θ_{i,k,C_2} is of index-level i and also of index-level k, i.e., this machine would appear in the set of machines with index-level of i as well as the set of machines with index-level of k. Notice that according to Lemma 14 these machines are already satisfied.

Lemma 17 *In Sol*, every machine (gap) is satisfied.*

Proof. We prove it through induction on the index-level of machines. We start with $i = 1$.

Consider machine $\theta_{\eta,1,+}$. We assume jobs $v_{i,2}$ and $\eta_{i',1/\delta-1,+}$ are on it. Then simple calculations show that

$$2f_{1/\delta}(1)x^{1/\delta} + 1 + g_{1/\delta-1}(1) = f_{1/\delta}(i)x^{1/\delta} + i + f_{1/\delta}(i')x^{1/\delta} + g_{1/\delta-1}(i').$$

According to Lemma 13, $f_{1/\delta}(1) = f_{1/\delta}(i) = f_{1/\delta}(i')$.

Since $i' \geq 1$, according to Observation 3 we have $g_{1/\delta-1}(i') \geq g_{1/\delta-1}(1)$. Meanwhile $i \geq 1$, thus $g_{1/\delta-1}(i') = g_{1/\delta-1}(1)$ and $i = 1$. Again, due to Observation 3 we have $i = i' = 1$. Thus $v_{1,2}$ and $\eta_{1,1/\delta-1,+}$ are on machine $\theta_{\eta,1,+}$, i.e., this machine is satisfied. Similarly we can prove that $v_{1,4}$ and $\eta_{1,1/\delta-1,-}$ are on machine $\theta_{\eta,1,-}$.

Consider machine $\theta_{1,j,+}$ for $1 \leq j \leq 1/\delta - 2$. We assume jobs $\eta_{i,j,+}$ and $\eta_{i',j+1,+}$ are on it. Then simple calculations show that

$$2\sum_{l=j+2}^{1/\delta} f_l(1)x^l + f_{j+1}(1)x^{j+1} + g_{j+1}(1) + g_j(1) = \sum_{l=j+1}^{1/\delta} f_l(i)x^l + \sum_{l=j+2}^{1/\delta} f_l(i')x^l + g_{j+1}(i') + g_j(i).$$

According to Lemma 13, we have

$$f_l(i) = f_l(1), \quad l = j+1, j+2, \cdots, 1/\delta,$$

$$f_l(i') = f_l(1), \quad l = j+2, j+3, \cdots, 1/\delta.$$

Thus $g_{j+1}(1) + g_j(1) = g_{j+1}(i') + g_j(i)$.

According to Observation 3, we have $g_j(i) \geq g_j(1)$ and $g_{j+1}(i') \geq g_{j+1}(1)$. Thus $g_j(i) = g_j(1)$, $g_{j+1}(i') = g_{j+1}(1)$. Again due to Observation 3 we have $i = i' = 1$, i.e., machine $\theta_{1,j,+}$ is satisfied.

Similarly we can prove that machine $\theta_{1,j,-}$ for $2 \leq j \leq 1/\delta - 2$ is also satisfied. For $j = 1$, recall that there is a slight difference between $\theta_{1,1,-}$ and $\theta_{1,1,+}$, we prove that machine $\theta_{1,1,-}$ is satisfied separately.

Consider $\theta_{1,1,-}$ and assume jobs $\eta_{i,1,-}$ and $\eta_{i',2,-}$ are on it. Then

$$2\sum_{l=3}^{1/\delta} \bar{f}_l(1)x^l + \bar{f}_2(1)x^2 + \bar{g}_2(1) + \bar{g}_1(1)x = \sum_{l=2}^{1/\delta} \bar{f}_l(i)x^l + \sum_{l=3}^{1/\delta} \bar{f}_l(i')x^l + \bar{g}_2(i') + \bar{g}_1(i)x.$$

According to Lemma 13, we have

$$\bar{f}_l(i) = \bar{f}_l(1), \quad l = 2, 3, \cdots, 1/\delta,$$

$$\bar{f}_l(i') = f_l(1), \quad l = 3, 4, \cdots, 1/\delta.$$

Thus $\bar{g}_2(1)+\bar{g}_1(1)x = \bar{g}_2(i')+\bar{g}_1(i)x$. Similarly due to observation 3 we have $\bar{g}_2(i') \geq \bar{g}_2(1)$, and $\bar{g}_1(i) \geq \bar{g}_1(1)$. Thus again we can prove $i = i' = 1$, which implies that machine $\theta_{1,1,-}$ is also satisfied.

Combining Lemma 15, Lemma 16 and Lemma 14, we have proved so far that each machine of index-level 1 is satisfied. We further show that indeed, all the variable and agent jobs of index-level 1 are on machines of index-level 1. To see why, see Figure 2 for an overview of the scheduling of jobs of index-level 1 (here Case 1 means $z_1 \in C_1$, while Case 2 means $\neg z_1 \in C_1$).

Suppose that for any $i < i_0 \leq n$, each machine of index-level i is satisfied and all the variable or agent jobs of index-level i are on machines of index-level i. We consider $i = i_0$.

According to Lemma 15 and Lemma 16, we know that machines $\theta_{i_0,k,+,C_1}$ (or $\theta_{i_0,k,-,C_1}$) for $k \in \{i_0, i_0+1, i_0+2\}$ and machines $\theta_{i_0,a,c}$, $\theta_{i_0,b,d}$, $\theta_{i_0,a,d}$, $\theta_{i_0,b,c}$ are satisfied.

Consider machine $\theta_{\eta,i_0,+}$ which is of index-level i_0. Again we may assume jobs $v_{i,2}$ and $\eta_{i',1/\delta-1,+}$ are on it, and the induction hypothesis implies that $i \geq i_0$, $i' \geq i_0$. Simple calculations show that

$$2f_{1/\delta}(i_0)x^{1/\delta} + i_0 + g_{1/\delta-1}(i_0) = f_{1/\delta}(i)x^{1/\delta} + i + f_{1/\delta}(i')x^{1/\delta} + g_{1/\delta-1}(i').$$

According to Lemma 13, $f_{1/\delta}(i_0) = f_{1/\delta}(i) = f_{1/\delta}(i')$. Since $i' \geq i_0$, according to Observation 3 we have $g_{1/\delta-1}(i') \geq g_{1/\delta-1}(i_0)$. Meanwhile $i \geq i_0$, thus $g_{1/\delta-1}(i') = g_{1/\delta-1}(i_0)$ and $i = i_0$. We can conclude that $i = i' = i_0$. So, $v_{i_0,2}$ and $\eta_{i_0,1/\delta-1,+}$ are on machine $\theta_{\eta,i_0,+}$, i.e., this machine is satisfied. Similarly we can prove that $v_{i_0,4}$ and $\eta_{i_0,1/\delta-1,-}$ are on machine $\theta_{\eta,i_0,-}$.

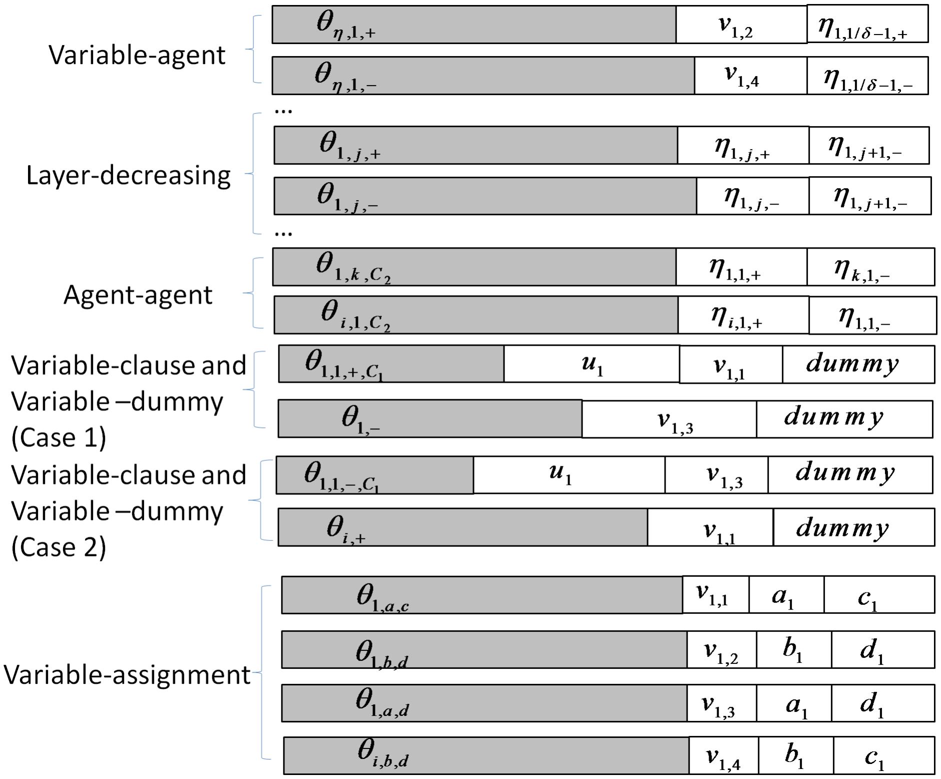

Figure 2: scheduling-of-indexlevel-1

Consider machine $\theta_{i_0,j,+}$ for $1 \le j \le 1/\delta - 2$. We assume jobs $\eta_{i,j,+}$ and $\eta_{i',j+1,+}$ are on it. Then simple calculations show that

$$2\sum_{l=j+2}^{1/\delta} f_l(i_0)x^l + f_{j+1}(i_0)x^{j+1} + g_{j+1}(i_0) + g_j(i_0) = \sum_{l=j+1}^{1/\delta} f_l(i)x^l + \sum_{l=j+2}^{1/\delta} f_l(i')x^l + g_{j+1}(i') + g_j(i).$$

According to Lemma 13, we have

$$f_l(i) = f_l(i_0), \quad l = j+1, j+2, \cdots, 1/\delta,$$

$$f_l(i') = f_l(i_0), \quad l = j+2, j+3, \cdots, 1/\delta.$$

Thus $g_{j+1}(i_0) + g_j(i_0) = g_{j+1}(i') + g_j(i)$.

According to the hypothesis we know $i, i' \ge i_0$. Due to Observation 3, we have $g_j(i) \ge g_j(i_0)$ and $g_{j+1}(i') \ge g_{j+1}(i_0)$. Thus $g_j(i) = g_j(i_0)$, $g_{j+1}(i') = g_{j+1}(i_0)$, which implies again that $i = i' = i_0$, i.e., machine $\theta_{i_0,j,+}$ is satisfied.

Similarly we can prove that machine $\theta_{i_0,j,-}$ for $1 \le j \le 1/\delta - 2$ is also satisfied (again we need to prove machine $\theta_{i_0,1,-}$ is satisfied separately, and the proof is actually the same as the case when $i_0 = 1$).

The above analysis shows that each machine of index-level i_0 is satisfied. Similar to the case when $i_0 = 1$, we can further show that all the variable and agent jobs of index-level i_0 are on machines of index-level i_0. $\square$

A machine is called truth benevolent if one of the following three conditions holds.

- For a variable-agent, layer-decreasing or agent-agent machine, the two jobs (variable or agent) on it are one true and one false.
- For a variable-clause machine, the variable and clause job on it are of the form (T, T), (F, F) or (T, F).
- For a variable-assignment machine, the variable and truth-assignment jobs on it are of the form (F, F, F) or (T, T, T).

We have the following lemma.

Lemma 18 *In Sol*, every machine of is truth benevolent.*

Proof. Consider a variable-agent, layer-decreasing or agent-agent machine. On each of these machines, the r-terms of the two (variable or agent) jobs should add up to $4r$ according to Lemma 11, thus the two jobs are one true and one false.

Consider a variable-clause machine. We check the r-terms of the clause, variable and dummy job. According to Lemma 11, there are three possibilities that the three r-terms add up to $11005r$, which are $r+10004r+1000r$, $3r+10002r+1000r$ and $r+10002r+1002r$, thus the variable and clause jobs are always of the form (T, T), (F, F) or (T, F).

Consider variable-assignment machines. We check the r-terms. Except for the huge job, the r-terms of the variable job, a_i or b_i, c_i or d_i should add up to $115r$ and thus there are only two possibilities, $r + 12r + 102r$ and $3r + 11r + 101r$, which implies that they are of the form (F, F, F) or (T, T, T). $\square$

Now we come to the proof of Lemma 9.

Proof. We assign values to variables according to the variable-assignment machines. For each i, consider the four machines, $\theta_{i,a,c}$, $\theta_{i,b,d}$, $\theta_{i,a,d}$ and $\theta_{i,b,c}$. Since the three jobs are (T, T, T) or (F, F, F), thus a_i^T is on the same machine with either c_i^T or d_i^T.

If a_i^T is scheduled with c_i^T, then the jobs on the two machines with $\theta_{i,a,c}$ and $\theta_{i,b,d}$ are $(v_{i,1}^T, a_i^T, c_i^T)$, $(v_{i,2}^T, b_i^T, d_i^T)$. We let variable z_i be false. Otherwise a_i^T is scheduled with d_i^T, and the jobs on the two machines with $\theta_{i,a,d}$ and $\theta_{i,b,c}$ are $(v_{i,3}^T, a_i^T, d_i^T)$ and $(v_{i,4}^T, b_i^T, c_i^T)$. We let variable z_i be true. We show that every clause is satisfied.

For each $c_j \in C_1$, there is one job u_j^T, and it should be scheduled with a true variable job. If it is $v_{i,1}^T$ where $i = j, j + 1$ or $j + 2$, then it turns out that z_i is true because otherwise $v_{i,1}^T$ is already scheduled with a_i^T and c_i^T. Notice that either machine $\theta_{j,i,+,C_1}$ or machine $\theta_{j,i,-,C_1}$ exists. Since $v_{i,1}$ is scheduled with u_j, machine $\theta_{j,i,-,C_1}$ does not exist because otherwise $v_{i,3}$, instead of $v_{i,1}$, is scheduled together with u_j on this machine. Thus the huge job $\theta_{j,i,+,C_1}$ exists, which means the positive literal z_i appears in c_j, thus c_j is satisfied. Otherwise it is $v^T_{i,3}$, then it turns out that z_i is false. As $v_{i,3}$ is scheduled with u_j, they are together with $\theta_{j,i,-,C_1}$, which means the negative literal $\neg z_i$ appears in c_j, and thus c_j is satisfied.

Consider each $(z_i \vee \neg z_k) \in C_2$. There is a huge job θ_{i,k,C_2}. As machine θ_{i,k,C_2} is satisfied and truth benevolent, $\eta_{i,1,+}$ and $\eta_{k,1,-}$ on this machine should be one true and one false according to Lemma 18.

Suppose on machine θ_{i,k,C_2}, $\eta_{i,1,+}$ is false and $\eta_{k,1,-}$ is true. Notice that there are two jobs, $\eta^T_{i,1,+}$ and $\eta^F_{i,1,+}$. Since $\eta^F_{i,1,+}$ is on machine θ_{i,k,C_2}, $\eta^T_{i,1,+}$ should be on machine $\theta_{i,1,+}$, and thus on this machine the other job is $\eta^F_{i,2,+}$. This further implies that $\eta^T_{i,2,+}$ and $\eta^F_{i,3,+}$ are on machine $\theta_{i,2,+}$. Carry on the above analysis until we reach machine $\theta_{i,1/\delta-2,+}$, and we know that $\eta^F_{i,1/\delta-1,+}$ is on this machine. Thus on machine $\theta_{\eta,i,+}$ the two jobs are $\eta^T_{i,1/\delta-1,+}$ and $v^F_{i,2}$. See Figure 3 for an illustration.

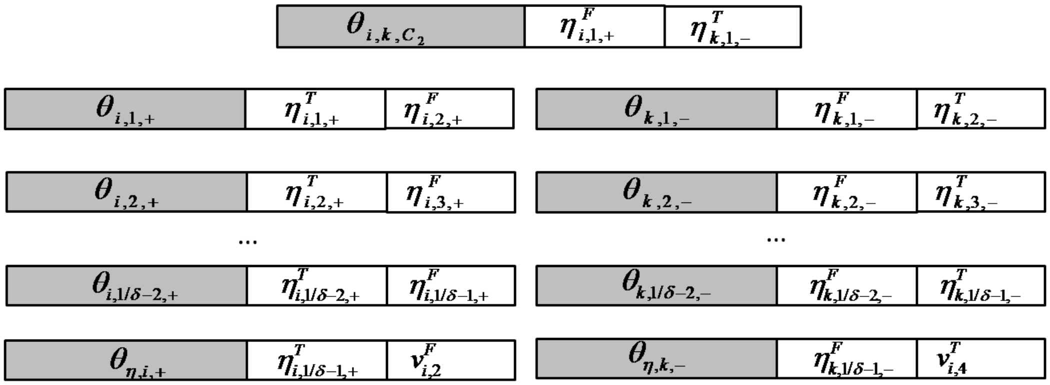

Figure 3: truth-assignment

Similarly, we can show that on machine $\theta_{\eta,k,-}$ the two jobs are $\eta^F_{k,1/\delta-1,-}$ and $v^T_{k,4}$. Thus, we can conclude that the variable z_k is false, because otherwise $v^T_{k,4}$ should be scheduled with b^T_k and c^T_k, which is a contradiction. So the clause $(z_i \vee \neg z_k)$ is satisfied.

Otherwise on machine θ_{i,k,C_2}, the two jobs are $\eta^T_{i,1,+}$ and $\eta^F_{k,1,-}$. Using the same argument as before we can show that on machine $\theta_{\eta,i,+}$, the job $\eta_{i,1/\delta-1,+}$ is false and the job $v_{i,2}$ is true, while on machine $\theta_{\eta,k,-}$, the job $\eta_{k,1/\delta-1,-}$ is true and the job $v_{k,4}$ is false. Thus, the variable z_i is true because otherwise $v^T_{i,2}$ should be scheduled with b^T_i and d^T_i, which is a contradiction. This implies that the clause $(z_i \vee \neg z_k)$ is satisfied. In both cases, every clause is satisfied, which means that I_{sat} is satisfiable. □

Recall that given any instance of the 3SAT' problem with n variables, for any $\delta > 0$ we construct a scheduling instance with $O(n/\delta)$ jobs such that it admits a feasible schedule of makespan $K = O(2^{3/\delta} n^{1+\delta})$ if and only if the given 3SAT' instance is satisfiable. Thus Theorem 5 (and also Theorem 1) follows directly. We prove Theorem 2.

Proof. Suppose the theorem fails, then there exists an exact algorithm for the restricted scheduling problem that runs in $2^{O(n^{1-\delta_0})}$ time for some $\delta_0 > 0$, then we may simply choose $\delta = \delta_0$ in our reduction. Since δ_0 is some fixed constant, the scheduling problem we construct contains $O(n)$ jobs with the processing time of each job bounded by $O(n^{1+\delta_0})$. Then we apply the scheduling algorithm to get an optimum solution, and it runs in $2^{O(n^{(1-\delta_0)(1+\delta_0)})}$, i.e., $2^{O(n^{1-\delta_0^2})}$ time. Through the makespan of this optimum solution, we can determine whether the given 3SAT' instance is satisfiable in $2^{O(n^{1-\delta_0^2})}$ time for some fixed $\delta_0 > 0$, resulting a contradiction. □

3 Scheduling on m Machines

We consider the scheduling problem on m ($m \geq 2$) identical machines and prove the following theorem.

Theorem 6 *Assuming ETH, there is no* $(1/\epsilon)^{o(\frac{m}{\log^2 m})}|I_{sche}|^{O(1)}$ *time FPTAS for* $Pm||C_{max}$.

Given the above theorem, Theorem 3 follows since otherwise, there exists a $(1/\epsilon)^{O(m^{1-\delta_0})}$ $|I|^{O(1)}$ time FPTAS for some $\delta_0 > 0$, and it runs in $(1/\epsilon)^{o(\frac{m}{\log^2 m})}|I_{sche}|^{O(1)}$ time, which is a contradiction.

To prove Theorem 6, given any 3SAT' instance I_{sat} with n variables, we construct a scheduling instance I_{sche} such that it admits an optimal solution with makespan $2^{O(\frac{n}{m}\log^2 m)}$ if and only if I_{sat} is satisfiable, then if the above theorem fails, we may apply the $(1/\epsilon)^{o(\frac{m}{\log^2 m})}|I_{sche}|^{O(1)}$ time PTAS for I_{sche} by setting $1/\epsilon = 2^{O(\frac{n}{m}\log^2 m)} + 1$. Simple calculations show that the optimal solution could be computed in $2^{\delta_m n}$ time where δ_m goes to 0 as m increases, and thus the satisfiability of the given 3SAT' instance could also be determined in $2^{\delta_m n}$ time, which is a contradiction.

We give an overview of the whole reduction. For simplicity throughout the remaining part of this section we assume the number of machines is $m+1$ with $m \geq 1$. Given a 3SAT' instance I_{sat}, we would transform it into an instance of the 3DM' problem, which is formally defined as follows:

- There are three disjoint sets of *elements* $W = \{w_i, \bar{w}_i | i = 1, \cdots, 3n\}$, $X = \{s_j, a_i | j = 1, \cdots, n', i = 1, \cdots, 3n\}$ and $Y = \{b_i | i = 1, \cdots, 3n\}$ where $n' \leq 3n$.
- There are three sets of *matches* $T_1 = \{(w_i), (\bar{w}_i) | i = 1, \cdots, 3n\}$, $T_2 \subseteq \{(w_i, s_j), (\bar{w}_i, s_j) | w_i \in W, s_j \in X\}$, $T_3 = \{(w_i, a_i, b_i), (\bar{w}_i, a_i, b_{\zeta(i)}) | i = 1, \cdots, 3n\}$ where ζ is defined as $\zeta(3k+1) = 3k+2$, $\zeta(3k+2) = 3k+3$ and $\zeta(3k+3) = 3k+1$ for $k = 1, \cdots, n$
- Either w_i or $\bar{w}_i$ appears in T_2, and it appears once. Every s_j appears at most three times in T_2.

A subset of $T = T_1 \cup T_2 \cup T_3$ is called an *exact cover* if every element of $W \cup X \cup Y$ appears exactly once in its matches.

We first show that, if I_{sat} contains n variables, then a 3DM' instance I_{3dm} in which there are $O(n)$ elements (indeed, $|W|$=6n) and $O(n)$ matches could be constructed such that I_{3dm} admits an exact cover if and only if I_{sat} is satisfiable. We then construct a scheduling instance I_{sche} which admits a feasible schedule of makespan $K = 2^{O(\frac{n}{m}\log^2 m)}$ if and only if I_{3dm} admits an exact cover, and thus if and only if I_{sat} is satisfiable.

We give a brief description of I_{sche}. Every match of T has a corresponding job (called as a match job) in I_{sche}. Recall that there are $m+1$ machines. If there is an exact cover, then I_{sche} admits a schedule of makespan K in which all the jobs corresponding to the matches of the exact cover are on the first m machines while other match jobs are on the last machine. Meanwhile, if I_{sche} admits a schedule of makespan K, then in this schedule there are m machines such that the matches correspond to the match jobs on them form an exact cover.

3.1 Modification of the 3SAT' Instance

Given a 3SAT' instance I_{sat}, we transform it into a 3DM' instance I_{3dm}, as we mention before..

Transforming I_{sat} to I'_{sat} Given a 3SAT' instance I_{sat} (with n variables), we can apply Tovey's method [26] to alter it into I'_{sat} by adding $O(n)$ variables and clauses such that the following properties are satisfied:

- Clauses of I'_{sat} could be divided into C_1 and C_2
  - Every variable appears once in C_1
  - Every (positive or negative) literal appears once in C_2
  - All the clauses of C_2 could be listed as $(z_{3k+1} \vee \neg z_{3k+2})$, $(z_{3k+2} \vee \neg z_{3k+3})$, $(z_{3k+3} \vee \neg z_{3k+1})$ for $k = 0, 1, \cdots, n-1$
- I'_{sat} is satisfiable if and only if I_{sat} is satisfiable

To see why, given I_{sat}, we replace each occurrence of a variable in I_{sat} with a new variable. Since every variable, say, z_i, appears three times, we can replace its three occurrences with $\hat{z}_{3i-1}$, $\hat{z}_{3i-1}$ and $\hat{z}_{3i}$ and meanwhile add $(z_{3i-2} \vee \neg z_{3i-1})$, $(z_{3i-1} \vee \neg z_{3i})$, $(z_{3i} \vee \neg z_{3i-2})$. Let C_2 be the set of all the newly added clauses and C_1 be the remaining clauses, then it is not difficult to verify that the modified instance I'_{sat} satisfies all the properties above.

Notice that there are $3n$ variables in I'_{sat}. For simplicity, we may further assume that n could be divided by m and let $n = qm$. (Indeed, if $n = qm + r$ with $0 < r < m$ with $q \geq 1$, we could add additionally $3(m-r)$ dummy variables, say, z^d_{3i+1}, z^d_{3i+2} and z^d_{3i+3} for $0 \leq i \leq m-r-1$, and meanwhile introduce $m-r$ dummy clauses in C_1 as

$(z^d_{3i+1} \vee z^d_{3i+2} \vee z^d_{3i+3})$, and $3(m-r)$ clauses in C_2 as $(z^d_{3i+1} \vee \neg z^d_{3i+1})$, $(z^d_{3i+2} \vee \neg z^d_{3i+3})$, $(z^d_{3i+3} \vee \neg z^d_{3i+1})$. It is not difficult to verify that these newly added variables and clauses do not change the satisfiability of the original instance.)

Transforming I'_{sat} to I_{3dm} We prove that, given I'_{sat} that contains $3n$ variables, a 3DM' instance I_{3dm} could be constructed such that I_{3dm} admits an exact cover if and only if I'_{sat} is satisfiable.

Construction of I_{3dm}. We construct two variable elements for each variable z_i, i.e., we construct w_i corresponding to z_i and $\bar{w}_i$ corresponding to $\neg z_i$. Let W be the set of them. It can be easily seen that $|W| = 6n$. We construct a clause element $s_j \in X$ for each $c_j \in C_1$.

Recall that all the clauses of C_2 could be listed as $(z_{3i+1} \vee \neg z_{3i+1})$, $(z_{3i+2} \vee \neg z_{3i+3})$, $(z_{3i+3} \vee \neg z_{3i+1})$ for $i = 0, 1, \cdots, n-1$. For every i, we construct $a_{3i+1}, a_{3i+2}, a_{3i+3} \in X$ and $b_{3i+1}, b_{3i+2}, b_{3i+3} \in Y$.

This completes the construction of elements and it can be easily seen that $|X| = 3n + m$, and $|Y| = 3n$. We construct matchings. For each variable z_i, we construct two matchings of T_1, namely (w_i) and $(\bar{w}_i)$.

For each clause $c_j \in C_1$, if the positive literal $z_i \in c_j$, then we construct $(w_i, s_j) \in T_2$. Else if the negative literal $\neg z_i \in c_j$, then we construct $(\bar{w}_i, s_j)$. Notice that c_j might contain two or three literals, thus two or three matchings of T_2 are constructed corresponding to it.

For each $0 \le i \le n-1$, 6 matchings of T_3 are constructed for the three clauses $(z_{3i+1} \vee \neg z_{3i+1})$, $(z_{3i+2} \vee \neg z_{3i+3})$ and $(z_{3i+3} \vee \neg z_{3i+1})$, namely $(w_{3i+1}, a_{3i+1}, b_{3i+1})$, $(w_{3i+2}, a_{3i+2}, b_{3i+2})$, $(w_{3i+3}, a_{3i+3}, b_{3i+3})$ and $(\bar{w}_{3i+1}, a_{3i+1}, b_{3i+2})$, $(\bar{w}_{3i+2}, a_{3i+2}, b_{3i+3})$, $(\bar{w}_{3i+3}, a_{3i+3}, b_{3i+1})$.

It can be easily seen that $|T_1| = 6n$, $|T_2| = 3n$, $|T_3| = 6n$. Thus in all, I_{3dm} contains $O(n)$ elements and matches.

The following part of this subsection is devoted to proving the following lemma. We remark that the proof is similar to the traditional proof showing that 3DM is NP-hard [5].

Lemma 19 *I'_{sat} is satisfiable if and only if I_{3dm} admits an exact cover.*

Proof. Suppose I'_{sat} is satisfiable, we choose matchings out of T to form an exact cover.

We know that for each $0 \le i \le n-1$, z_{3i+1}, z_{3i+2} and z_{3i+3} are either all true or all false. If they are all true, then we choose $(\bar{w}_{3i+1}, a_{3i+1}, b_{3i+2})$, $(\bar{w}_{3i+2}, a_{3i+2}, b_{3i+3})$, $(\bar{w}_{3i+3}, a_{3i+3}, b_{3i+1})$. Otherwise they are all false, and $(w_{3i+1}, a_{3i+1}, b_{3i+1})$, $(w_{3i+2}, a_{3i+2}, b_{3i+2})$, $(w_{3i+3}, a_{3i+3}, b_{3i+3})$ are chosen instead.

Now every element of Y appears exactly once in the matches we choose currently. Since each clause $c_j \in C_1$ is satisfied, it is satisfied by at least one variable. We choose the variable that leads to the satisfaction of c_j (if there are multiple such variables, we choose arbitrarily one). Suppose this variable is z_i. If z_i is true, then we know the positive literal $z_i \in c_j$. According to our construction $(w_i, s_j) \in T_2$ and we choose it. Otherwise z_i is false, and the negative literal $\neg z_i \in c_j$. Again it follows that $(\bar{w}_i, s_j) \in T_2$ and we choose it.

Consider the matches we have chosen so far. Every element of X and Y appears exactly once in these matchings. Moreover, each element of W appears at most once in these matchings. To see why, notice that if we choose $(w_i, s_j) \in T_2$, for example, then z_i is true and we do not choose matchings of T_3 that contain w_i. Finally, we choose matchings of T_1 to enforce that every element of W appears exactly once.

On the contrary, suppose there exists an exact cover of I_{3dm}, we prove that I'_{sat} is satisfiable. Consider elements of X and Y. For each $0 \le i \le n-1$, to ensure that a_{3i+1}, b_{3i+1}, a_{3i+2}, b_{3i+2} and a_{3i+3}, b_{3i+3} appear once respectively, in the exact cover T' we have to choose either $(\bar{w}_{3i+1}, a_{3i+1}, b_{3i+2})$, $(\bar{w}_{3i+2}, a_{3i+2}, b_{3i+3})$, $(\bar{w}_{3i+3}, a_{3i+3}, b_{3i+1})$, or choose $(w_{3i+1}, a_{3i+1}, b_{3i+1})$, $(w_{3i+2}, a_{3i+2}, b_{3i+2})$, $(w_{3i+3}, a_{3i+3}, b_{3i+3})$.

If $(\bar{w}_{3i+1}, a_{3i+1}, b_{3i+2})$, $(\bar{w}_{3i+2}, a_{3i+2}, b_{3i+3})$, $(\bar{w}_{3i+3}, a_{3i+3}, b_{3i+1})$ are in T', we set z_{3i+1}, z_{3i+2} and z_{3i+3} to be true. Otherwise we set z_{3i+1}, z_{3i+2} and z_{3i+3} to be false. It can be easily seen that every clause of C_2 is satisfied.

We consider $c_j \in C_1$. Notice that $s_j \in X$ appears once in T'. Suppose the match containing s_j is (s_j, w_i) for some i, then it follows that the positive literal $z_i \in c_j$. The fact that w_i also appears once implies that $\bar{w}_i$ appears in $T' \cap T_3$, and thus variable z_i is true and c_j is satisfied.

Otherwise the matching containing s_j is $(s_j, \bar{w}_i)$ for some i, then similar arguments show that the negative literal $\neg z_i \in c_j$ and variable z_i is false. Again c_j is satisfied. □

3.2 Defining the Functions f and g Based on Partitioning Matches

Notice that there are one-element, two-element and three-element matches in I_{3dm}, throughout this section we may use (w_i, x_j, y_k) to represent any match of $T = T_1 \cup T_2 \cup T_3$, where $x_j = y_k = \emptyset$ if it represents a one-element match, and $y_k = \emptyset$ if it represents a two-element match.

Recall that $n = qm$ for some integer q. We divide the set W equally into m subsets, with $W_k = \{w_i, \bar{w}_i | 3kq + 1 \le i \le 3kq + 3q\}$ for $0 \le k \le m-1$. We aim to construct a scheduling instance in which there is one job for every match, and the goal of this subsection is to construct the functions f and g through which the processing time of the job corresponding to (w_i, x_j, y_k) is defined as $g(w_i)\alpha^{f(w_i)} + g(x_j)\alpha^{f(x_j)} + g(y_k)\alpha^{f(y_k)}$ where $\alpha = m^{O(1)}$ and $g(w_i), g(x_j), g(y_k) < \alpha$. Given any exact cover (if it exists), a schedule for the constructed scheduling instance is called a natural schedule (corresponding to the exact cover) if jobs corresponding to matches of this exact cover that contain $w_i \in W_k$ or $\bar{w}_i \in W_k$ are on machine k. We try to design f and g such that the following properties hold for a natural schedule,

1. $f(\emptyset) = g(\emptyset) = 0$, otherwise $f(x_j), g(x_j), f(y_k), g(y_k) \in \mathbb{Z}^+$.

2. For any (w_i, x_j, y_k) and $(w_{i'}, x_{j'}, y_{k'})$ from the exact cover, if $f(w_i) = f(w_{i'}) \neq 0$ (or $f(x_j) = f(x_{j'}) \neq 0$ or $f(y_k) = f(y_{k'}) \neq 0$), then the two corresponding jobs should be on different machines in the natural schedule.

3. The largest value returned by f (denoted as $|f|$) is as small as possible.

The second property above actually ensures that if jobs corresponding to matches of an exact cover are on the same machine according to the corresponding natural schedule, then their processing times contain different α^i terms, and it is a key property through which a natural schedule of makespan K could be constructed from the given exact cover for I_{3dm}.

The following part of this subsection is devoted to designing the functions f and g satisfying the properties above. Obviously we can simply define f so that it takes a distinct value for every element, and then $|f| = O(n)$, which is too large. In the following part, we will define f such that $|f| = O(n/m \log m)$.

We construct a bipartite graph $G = (V^w \cup V^s, E)$ in the following way. There are m vertices in V^w, with a slight abuse of notations we denote them as W_k for $0 \leq k \leq m-1$. There are $|C_1| \leq 3n$ vertices in V^s, and we denote them as s_j for $1 \leq j \leq |C_1|$. There is an edge between W_k and s_j if $(w_i, s_j) \in T_2$ or $(\bar{w}_i, s_j) \in T_2$ where $w_i, \bar{w}_i \in W_k$.

It can be easily verified that the degree of every W_k is at most $3q$, while the degree of every s_j is at most 3.

We define f in the following way. According to the special structure of T_1 and T_3, we first let $f(b_{3kq+i}) = i$, $f(a_{3kq+i}) = 3q + i$, $f(w_{3kq+i}) = 6q + i$ and $f(\bar{w}_{3kq+i}) = 9q + i$. It can be easily seen that jobs corresponding to one-element and three-element matches of the exact cover do not share the same α^i term for $1 \leq i \leq 12q$ if they are on machine k (according to the natural schedule). For jobs corresponding to two-element matches, the following property can ensure that they do not share the same α^i term if they are on the same machine: for any s_j and $s_{j'}$ which are both connected to some W_k, $f(s_j) \neq f(s_{j'})$. We associate the value of each distinct $f(s_j)$ with a specific color. The problem becomes that, can we draw each vertex of V^s with a color so that for any $0 \leq k \leq m-1$, all the s_j connected to W_k are drawn with different colors.

Obviously we can draw every s_j with a distinct color and this yields a solution with $O(n)$ colors. We provide a natural greedy algorithm which uses $O(n/m \log m)$ colors, and the function f is defined accordingly.

Consider the following method of coloring. We sequence s_j in an arbitrary order. Suppose we have used t different colors and currently we pick a new color $t+1$. We take the first uncolored s_j in the sequence and color it with $t+1$. Then we delete vertices in the sequence sharing the same neighbors with s_j (since they could not be colored with $t+1$). Now we pick the first remaining vertex in the sequence and color it. We carry on this procedure until we can not color any more, then we pick a new color $t+2$.

To analyze the number of colors used by the above algorithm, we have the following lemma.

Lemma 20 *If currently there are $h \leq |C_1|$ vertices uncolored, then using a new color we could color at least $\lceil \frac{h}{9q} \rceil$ vertices.*

Proof. Suppose s_j is colored with color t. Then according to the greedy algorithm, vertices sharing the same neighbors with it should be deleted. Recall that the degree of

s_j is at most 3, while the degree of W_k $(0 \le k \le m-1)$ is at most $3q$. Thus we delete at most $9q-1$ vertices and the lemma follows directly. □

Let ϕ_k be the number of vertices that are not colored when k different colors are used, then $\phi_0 = |C_1| \le 3n$ and it follows directly that $\phi_{k+1} \le \phi_k - \lceil \frac{\phi_k}{9q} \rceil$.

Thus $\phi_{k+1} \le \phi_k(1-\frac{1}{9q})$. Meanwhile $\phi_{k+1} \le \phi_k - 1$, so the integer sequence goes to 0. We assume $|C_1| \ge 9q$, otherwise we use at most $9q = O(n/m)$ colors. Let k_0 be the largest index such that $\phi_{k_0} \ge 9q$, then it follows that $|C_1|(1-\frac{1}{9q})^{k_0} \ge 9q$, and simple calculations show that $k_0 \le \lceil \log_{1-\frac{1}{9q}} \frac{9q}{m_1} \rceil \le 9q \ln m$. Thus in all the greedy algorithm uses $\tau \le 9q \ln m + 9q$ different colors.

Let $\chi(j) \le \tau$ be the color of s_j. We define the function f in the following way.

- $f(b_{3kq+i}) = i$, $f(a_{3kq+i}) = 3q+i$.
- $f(w_{3kq+i}) = 6q+i$, $f(\bar{w}_{3kq+i}) = 9q+i$.
- $f(s_j) = 12q + \chi(j)$.

As we mention before, a job corresponding to a match, say, (w_i, x_j, y_k) has a processing time of $g(w_i)\alpha^{w_i} + g(x_j)\alpha^{f(x_j)} + g(y_k)\alpha^{f(y_k)}$ where $\alpha = m^{O(1)}$. The function f indicates the index of the exponential, while the function g indicates the coefficient. According to our previous discussion if we put jobs corresponding to matches that contain w_{3kq+i} or $\bar{w}_{3kq+i}$ onto machine k (as the natural schedule indicates), then all the jobs on machine k have different α^i terms.

We provide the definition of g now.

- $g(w_{3kq+i}) = g(\bar{w}_{3kq+i}) = g(a_{3kq+i}) = g(b_{3kq+i}) = m+k$.
- Sort vertices with the same color in an arbitrary way. Suppose s_j is colored with color t and is the l-th vertex in the sequence, then $g(s_j) = m+l-1$.

We have the following simple observation on g.

Lemma 21 *$g(s_j) < 2m$ for $1 \le j \le |C_1|$.*

Proof. Suppose the lemma fails, then there are at least $m+1$ vertices coloring with the same color. Notice that every s_j is connected with some W_k for some $0 \le k \le m-1$, among these $m+1$ vertices there are two of them connecting to the same W_k, implying that they should be colored with different colors, which is a contradiction. □

3.3 Construction of the Scheduling Instance

Let $\alpha = 6m^4 + 6m^3 + 6m^2$ be the base. Recall that $n = qm$, $\tau = O(n/m \log m)$. With a slight abuse of notation we let $|f| = 12q + \tau = O(n/m \log m)$. We construct a scheduling instance I_{sche} such that it admits an optimal solution of makespan $6m^3 \sum_{i=1}^{|f|} \alpha^i = 2^{O(n/m \log^2 m)}$ if and only if I_{3dm} admits an exact cover.

We construct three kinds of jobs, a match job for every match, a cover job for every element and dummy jobs.

For every match (w, x, y), we construct a job with processing time $g(w)\alpha^{f(w)} + g(x)\alpha^{f(x)} + g(y)\alpha^{f(y)}$ where (w, x, y) may represent (w_i) or $(\bar{w}_i)$ (in this case we take $g(x) = g(y) = 0$), or represent (w_i, s_j) or $(\bar{w}_i, s_j)$ (in this case we take $g(y) = 0$), or represent (w_i, a_i, b_i) or $(\bar{w}_i, a_i, b_{\zeta(i)})$.

For every element η, we construct a job with processing time $(6m^3 - g(\eta))\alpha^{f(\eta)}$ where η may represent w_i, $\bar{w}_i$, s_j, a_i or b_i.

We construct dummy jobs. For each color we would construct several dummy jobs, and we also construct a huge dummy job.

According to Lemma 21, we suppose there are $l_t \le m$ vertices colored with color t. If $l_t < m$, we then construct $m - l_t$ dummy jobs, each of which has a processing time of $6m^3\alpha^{12q+t}$. We call these dummy jobs as dummy jobs of color t.

Recall that there are $m+1$ machines, we construct a huge dummy job whose processing time equals to $6m^3(m+1)\sum_{i=1}^{9q+\tau} \alpha^i$ minus the total processing time of all the jobs we construct before. It follows directly that if there exists a feasible solution for I_{sche} whose makespan is no more than $6m^3 \sum_{i=1}^{|f|} \alpha^i$, the load of every machine is $6m^3 \sum_{i=1}^{|f|} \alpha^i$.

3.4 From 3DM' to Scheduling

For simplicity we index the $m+1$ machines as machine 0 to machine m. We provide a feasible solution of I_{sche} with makespan $6m^3 \sum_{i=1}^{|f|} \alpha^i$.

Given the exact cover of I_{3dm}, we take out the jobs corresponding to the matches of the exact cover. We put jobs corresponding to the matches that contain w_{3kq+i} or $\bar{w}_{3kq+i}$ onto machine k for $0 \le k \le m-1$. According to our definition of f, each job on machine k contains distinct α^i terms.

Let L_k be total processing time of jobs on machine k. It is not difficult to verify that for $1 \le i \le 12q$, the coefficient of α^i term is $g(k) = m + k$. By adding the corresponding cover job (with processing time $(6m^3 - g(k))\alpha^i$), it becomes $6m^3\alpha^i$.

For the α^i term with $12q + 1 \le i \le 12q + \tau$, the coefficient is either $g(s_j)$ or 0. If it is $g(s_j)$, we add a corresponding cover job, otherwise we add a dummy job of processing time $6m^3\alpha^i$. In both cases the it becomes $6m^3\alpha^i$. Recall that for color t, there are l_t cover jobs and $m - l_t$ dummy jobs, so we can always add jobs in the above way.

Now the loads of machine 0 to machine $m-1$ are all $6m^3 \sum_{i=1}^{|f|} \alpha^i$, implying that if we put all the remaining jobs to machine m, its load is also $6m^3 \sum_{i=1}^{|f|} \alpha^i$.

3.5 From Scheduling to 3DM'

Given a solution for I_{sche} in which the load of each machine is $6m^3 \sum_{i=1}^{|f|} \alpha^i$, we construct a an exact cover. We assume that the huge job is on machine m, and the following lemma follows.

Lemma 22 *For jobs on machine* 0 *to machine* $m-1$*, the coefficients of their* α^i *terms* $(1 \leq i \leq |f|)$ *add up to* $6m^3\alpha^i$*.*

Proof. Notice that the huge job is on machine m, we prove that, except for this job, the sum of all the coefficients of α^i term is bounded by $\alpha - 1$. This would be enough to ensure that there is no "carry-over"from α^i to α^{i+1} when we compute the sum of the processing times of the jobs on machine k for $0 \leq k \leq m-1$, and the lemma follows directly.

Consider the α^i term for $1 \leq i \leq 3q$. For each $0 \leq k \leq m-1$, since b_{3kq+i} appear twice in matches, hence the sum of all the coefficients of α^i terms is bounded by $\sum_{k=0}^{m-1}[6m^3 - (m+k)] + 2\sum_{k=0}^{m-1}(m+k) \leq 6m^4 + 3m^2 < \alpha$.

The sum of coefficients of α^i term for $i > 3q$ could be verified in the same way. □

Lemma 23 *For machine* 0 *to machine* $m-1$*, the term* $6m^3\alpha^i$ *in the sum is contributed by either a dummy job, or a cover and match job.*

Proof. If the coefficient $6m^3$ of the α^i term is contributed by only one job, then obviously it is a dummy job. Otherwise it is contributed by two or more jobs, and among these jobs there could be at most one cover job since the coefficient of a cover job is at least $6m^3 - (m+k) > 5m^3$. Meanwhile there should be at least one cover job, otherwise all the coefficients of match jobs add up to $3\sum_{k=0}^{m-1}(m+k) < 6m^3$ (every element appear at most three times in matches), which is a contradiction.

Given that the coefficient $6m^3$ is contributed by exactly one cover job, the coefficients of the remaining jobs add up to a value in $(m, 2m)$, implying that it is a match job. □

Lemma 24 *All the cover jobs are on machine* 0 *to machine* $m-1$*.*

Proof. Consider jobs with nonzero α^i terms for $1 \leq i \leq 12q$. There are m cover jobs whose coefficients are $6m^3 - m$, $6m^3 - (m+1)$, $\cdots$, $6m^3 - (2m-1)$, and no dummy jobs. According to Lemma 23 these cover jobs must be on machine 0 to machine $m-1$, one for each.

Consider jobs with nonzero α^{12q+t} terms for $1 \leq t \leq \tau$. There are l_t cover jobs whose coefficients are $6m^3 - m$, $\cdots$, $6m^3 - (m + l_t - 1)$, and $m - l_t$ dummy jobs, again due to Lemma 23 we know they are on machine 0 to machine $m-1$, one for each. □

For any $i \leq |f|$, if the coefficient $6m^3$ of the α^i term is contributed by a cover and a match job, then the match corresponding to the match job contains the element corresponding to the cover job. Furthermore, there is a cover job for every element, thus if we take the matches corresponding to the match jobs on machine 0 to machine $m-1$, they form an exact cover of I_{3dm}.

This completes the proof of Theorem 6. Simple calculations show that the input length of the scheduling instance we construct is $|I_{sche}| \leq O(\frac{n \log^3 m}{m})$, and from the same reduction Theorem 4 also follows. On the other hand, the traditional dynamic programming algorithm for $Pm||C_{max}$ runs in $2^{O(\sqrt{m|I|\log m}+m\log|I|)}$ time, as is shown in the next section.

4 Dynamic Programming for $Pm||C_{max}$

We show in this section that the traditional dynamic programming algorithm for the scheduling problem runs in $2^{O(\sqrt{m|I|\log m}+m\log|I|)}$ time.

Consider the dynamic programming algorithm for the scheduling problem. Suppose jobs are sorted beforehand as $p_1 \leq p_2 \leq \cdots \leq p_n$. We use a vector $(k, t_1, t_2, \cdots, t_m)$ to represent a schedule for the first k jobs where the load of machine i (i.e., total processing times of jobs on machine i) is t_i. Let ST_k be the set of all these vectors that correspond to some schedules. We could determine ST_k iteratively in the following way.

Let $ST_0 = (0, 0, 0, \cdots, 0)$. For $k \geq 1$, $(k, t_1, t_2, \cdots, t_m) \in ST_k$ if there exists some $(k-1, t'_1, t'_2, \cdots, t'_m) \in ST_{k-1}$ such that for some $1 \leq i \leq m$, $t_i = t'_i + p_k$, and $t_j = t'_j$ for $j \neq i$. Since each vector of ST_{k-1} can give rise to at most m different vectors of ST_k, the computation of the set ST_k thus takes $O(m|ST_{k-1}|)$ time. Meanwhile, once ST_n is determined, we check each vector of it and select the one whose makespan is minimized, which also takes $O(m|ST_n|)$ time. After the desired vector is chosen, we may need to backtrack to determine how jobs are scheduled on each machine, and this would take $O(n)$ time.

Thus, the overall running time of the dynamic programming algorithm mainly depends on the size of the set $|ST_k|$ for $1 \leq k \leq n$. We have the following lemma.

Lemma 25

$$|ST_k| \leq 2^{O(\sqrt{m|I|\log m}+m\log|I|)}.$$

Proof. Notice that each vector of ST_k corresponds to some schedule. Let $J_1, J_2 \cdots, J_k$ be the first k jobs with processing times $1 \leq p_1 \leq p_2 \leq \cdots \leq p_k$ and $\lambda_k = \log_2 \prod_{i=1}^k p_i$. Notice that such an indexing of jobs is only used in the proof, while in the dynamic programming jobs are in arbitrary order. There are three possibilities.

Case 1: $\log_2 p_1 \geq \sqrt{\lambda_k \log_2 m/m}$. Since each vector in ST_k corresponds to a schedule, we consider all possible assignments of the k jobs. Each job could be assigned to m machines, thus there are at most $m^k = 2^{k \log_2 m}$ different assignments for k different jobs.

Since $1 \leq p_1 \leq p_2 \leq \cdots \leq p_k$, we have

$$\lambda_k = \sum_{i=1}^{k} \log_2 p_i \geq k\sqrt{\lambda_k \log_2 m/m},$$

thus $k\log_2(m+1) \le \sqrt{\lambda_k m \log_2 m}$, which implies that

$$|ST_k| \le 2^{k\log_2 m} \le 2^{\sqrt{\lambda_k m \log_2 m}}.$$

Case 2: $\log_2 p_k \le \sqrt{\lambda_k \log_2 m/m}$. Consider any vector of ST_k, say, $(k, t_1, t_2, \cdots, t_m)$. As $t_i \le kp_k$, there are at most $(kp_k)^m = 2^{m(\log_2 k + \log_2 p_k)}$ different vectors. It can be easily seen that

$$|ST_k| \le 2^{m(\log_2 k + \log_2 p_k)} \le 2^{\sqrt{\lambda_k m \log_2 m} + m\log_2 k}.$$

Case 3: There exists some $1 \le k_0 \le k-1$ such that $\log_2 p_{k_0} \le \sqrt{\lambda_k \log_2 m/m}$ and $\log_2 p_{k_0+1} \ge \sqrt{\lambda_k \log_2 m/m}$.

Notice that each vector of ST_k corresponds to some schedule. Given $(k, t_1, t_2, \cdots, t_m) \in ST_k$, we may let G_i be the set of jobs on machine i. Group G_i can be split into two subgroups, i.e., jobs belonging to the set $\{J_1, J_2, \cdots, J_{k_0}\} \cap G_i$ and the set $\{J_{k_0+1}, J_{k_0+2}, \cdots, J_k\} \cap G_i$. Let $t_i^{(1)}$ be the total processing time of jobs in the former subgroup and $t_i^{(2)}$ be the total processing time of jobs in the latter subgroup. Then the vector $(t_1, t_2, \cdots, t_m)$ can be expressed as the sum of two vectors

$$(t_1, t_2, \cdots, t_m) = (t_1^{(1)}, t_2^{(1)}, \cdots, t_m^{(1)}) + (t_1^{(2)}, t_2^{(2)}, \cdots, t_m^{(2)}).$$

Let $ST_k^{(1)}$ and $ST_k^{(2)}$ be the sets of all possible vectors $(t_1^{(1)}, t_2^{(1)}, \cdots, t_m^{(1)})$ and $(t_1^{(2)}, t_2^{(2)}, \cdots, t_m^{(2)})$ respectively, then we know $|ST_k| \le |ST_k^{(1)}| \times |ST_k^{(2)}|$. Consider each vector of $ST_k^{(1)}$, it corresponds to some feasible schedule of jobs 1 to k_0 over machines. Since $t_i^{(1)} \le k_0 p_{k_0}$ and $\log_2 p_{k_0} \le \sqrt{\lambda_k \log_2 m/m}$, we have

$$|ST_k^{(1)}| \le (k_0 p_{k_0})^m \le 2^{m\log_2 k_0 + \sqrt{\lambda_k m \log_2 m}}.$$

Consider each vector of $ST_k^{(2)}$, it corresponds to some feasible schedule of jobs k_0+1 to k over machines. To assign $k-k_0$ different jobs to m machines, there are at most $m^{k-k_0} = 2^{(k-k_0)\log_2 m}$ different assignments. Since $\log_2 p_{k_0+1} \ge \sqrt{\lambda_k \log_2 m/m}$, we have

$$\lambda_k \ge \sum_{i=k_0+1}^{k} \log_2 p_i \ge (k-k_0)\sqrt{\lambda_k \log_2 m/m},$$

thus $(k-k_0)\log_2 m \le \sqrt{\lambda_k m \log_2 m}$, which implies that

$$|ST_k^{(2)}| \le 2^{(k-k_0)\log_2 m} \le 2^{\sqrt{\lambda_k m \log_2 m}}.$$

Thus,

$$|ST_k| \le |ST_k^{(1)}| \times |ST_k^{(2)}| \le 2^{m\log_2 k_0 + 2\sqrt{\lambda_k m \log_2 m}}.$$

In any of the above three cases, we always have

$$|ST_k| \le 2^{O(\sqrt{m|I|\log m} + m\log|I|)}.$$

□

5 Conclusion

We provide lower bounds for exact and approximation algorithms of scheduling problems under the Exponential Time Hypothesis. We show that, the traditional dynamic programming for $P||C_{max}$ and the best known FPTAS for $Pm||C_{max}$ are essentially the best possible, while the lower bound of $2^{O((1/\epsilon)^{1-\delta})}\ |I|^{O(1)}$ for any $\delta > 0$ and upper bound of $2^{O(1/\epsilon^2 \log^3(1/\epsilon))} + O(n \log n)$ for the running time of the PTAS for $P||C_{max}$ leave some room for improvement. Furthermore, we conjecture that there is an EPTAS with running time $2^{O(1/\epsilon \log^c(1/\epsilon))}|I|^{O(1)}$ for $P||C_{max}$ where c is some constant. Finally, we believe that the techniques could be useful also for other scheduling and packing problems.

References

[1] N. Alon, Y. Azar, G.J. Woeginger, and T. Yadid: Approximation schemes for scheduling on parallel machines, *Journal on Scheduling*, 1, 1998, 55–66.

[2] C. Calabro, R. Impagliazzo, and R. Paturi: A duality between clause width and clause density for sat. *IEEE Conference on Computational Complexity (CCC 2006)*, 252–260.

[3] J. Chen, X. Huang, I. Kanj, and G. Xia: On the computational hardness based on linear FPT-reductions. *Journal of Combinatorial Optimization*, 11, 2006, 231-247.

[4] A. Fishkin, K. Jansen, and M. Mastrolilli: Grouping techniques for scheduling problems: simpler and faster, *Algorithmica*, 51, 2008, 183–199.

[5] M.R. Garey, and D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, chapter 34. W. H. Freeman and Company, 1979.

[6] T. Hertli: 3-Sat faster and simpler - unique-Sat bounds for PPSZ hold in general, 52th IEEE Symposium on Foundations of Computer Science (FOCS 2011), 277–284.

[7] D.S. Hochbaum and D.B. Shmoys: Using dual approximation algorithms for scheduling problems: practical and theoretical results, *Journal of the ACM*, 34, 1987, 144–162.

[8] D.S. Hochbaum: Various notions of approximations: good, better, best, and more, in: *Approximation Algorithms for NP-Hard Problems*, D.S. Hochbaum, ed., Prentice Hall, 1977, 346–398.

[9] E. Horowitz and S. Sahni: Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23, 1976, 317–327.

[10] R. Impagliazzo, R. Paturi, and F. Zane: Which problems have strongly exponential complexity? *Journal of Computer and System Science*, 63.4, 2001, 512–530.

[11] K. Jansen: An EPTAS for Scheduling Jobs on Uniform Processors: Using an MILP Relaxation with a Constant Number of Integral Variables. *SIAM Journal on Discrete Mathematics* 24(2), 2010, 457–485.

[12] K. Jansen, F. Lang, and K. Lang: Bounding the running time of algorithms for scheduling and packing problems. University of Kiel, Technical Report 1302 (2013).

[13] K. Jansen and M. Mastrolilli: Scheduling unrelated parallel machines: linear programming strikes back. University of Kiel, Technical Report 1004 (2010).

[14] K. Jansen and L. Porkolab: Improved approximation schemes for scheduling unrelated parallel machines. *Mathematics of Operations Research* 26(2), 2001, 324–338.

[15] K. Jansen and C. Robenek: Scheduling jobs on identical and uniform processors revisited. *Workshop on Approximation and Online Algorithms (WAOA 2011)*, 109–122.

[16] A. Kulik and H. Shachnai: There is no EPTAS for two-dimensional knapsack. *Information Processing Letters*, 110, 2010, 707–710.

[17] J. K. Lenstra, D. B. Shmoys, and Eva Tardos: Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programing*, 46, 1980, 259–271.

[18] C. Lente, M. Liedloff, A. Soukhal, and V. T'Kindt: On an extension of the sort and search method with application to scheduling theory. submitted to *Theoretical Computer Science.*

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

[20] D. Lokshtanov, D. Marx, and S. Saurabh: Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS* 105, 2011, 41–72.

[21] D. Marx: On the optimality of planar and geometric approximation schemes. *IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, 338–348.

[22] D. Marx: Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1), 2008, 60–78.

[23] T. O'Neil: Sub-exponential algorithms for 0/1 knapsack and bin packing. Unpublished Manuscript.

[24] T. O'Neil and S. Kerlin: A simple $2^{O(\sqrt{x})}$ algorithm for partition and subset sum. *International Conference on Foundations of Computer Science (FCS 2010)*, 55–58.

[25] M. Patrascu and R. Williams: On the possibility of faster SAT algorithms. *ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, 1065–1075.

[26] C.A. Tovey: A simplified satisfiability problem, *Discrete Applied Mathematics*, 8, 1984, 85–89.

[27] G. Woeginger: Exact algorithms for NP-hard problems: A survey, in M. Junger, G. Reinelt, G. Rinaldi (Eds): Combinatorial Optimization–Eureka! You shrink!, LNCS 2570 (2003), 185–207.