

TEL AVIV UNIVERSITY
THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
Department of Electrical Engineering - Systems

SCHEDULING WITH CONFLICTS

Thesis submitted toward the degree of
Master of Science in Electrical Engineering
in Tel-Aviv University

by

Lotem Kaplan

May 2007

TEL AVIV UNIVERSITY
THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
Department of Electrical Engineering - Systems

SCHEDULING WITH CONFLICTS

Thesis submitted toward the degree of
Master of Science in Electrical Engineering
in Tel-Aviv University

by

Lotem Kaplan

This research was carried out at Tel-Aviv University
in the Department of Electrical Engineering - Systems,
Faculty of Engineering
under the supervision of Prof. Dana Ron and Dr. Guy Even

May 2007

Contents

1	Introduction	1
1.1	Previous results	3
1.2	Our results	4
1.3	Other Related Work	5
2	Preliminaries	7
2.1	The model	7
2.2	Additional terminology	8
3	m Machines & Arbitrary Processing Times	9
3.1	Analysis of GME's makespan	10
3.2	Tightness of the approximation of the makespan	11
4	Two Machines & $p_j \in \{1, 2, 3\}$	13
4.1	Unit jobs	13
4.2	An Optimal Algorithm for Processing Times $p_j \in \{1, 2\}$	13
4.2.1	Intuition	13
4.2.2	Auxiliary graph	14
4.2.3	Scheduling algorithm	15
4.2.4	Correctness	16
4.2.5	On the relation to b -matching	19
4.3	$\frac{4}{3}$ -Approximation for Processing Times $p_j \in \{1, 2, 3\}$	19

4.3.1	Intuition	19
4.3.2	Auxiliary graph	20
4.3.3	Scheduling algorithm	20
4.3.4	Correctness	22
4.3.5	Tightness	26
5	A Reduction to k-Set-Cover	27
5.1	Unit Jobs	27
5.2	Arbitrary Processing Times	28
	Bibliography	29

List of Figures

1.1	Scheduling with conflicts in a wireless network. Suppose there are three messages: (1) $M_{1,1}$ from i_1 to j_1 , (2) $M_{1,2}$ from i_1 to j_2 , and (3) $M_{2,3}$ from i_2 to j_3 . Message $M_{1,2}$ conflicts both with $M_{1,1}$ (same station) and $M_{2,3}$ (disturbance from i_2). Messages $M_{1,1}$ and $M_{2,3}$ do not conflict.	2
3.1	Tight constructions for the greedy algorithm. (A) A construction for the case that m is even. (B) A construction for odd values of m . The m cliques are arranged in a cycle.	12
4.1	(A) An agreement graph over three jobs j, u, v . The processing times are $p_j = 1$ and $p_u = p_v = 2$. (B) The corresponding auxiliary graph. Job slices of type-a are depicted using a filled circle while job slices of type-b are depicted using an unfilled circle. Unit jobs lack a circle.	15
4.2	An unfilled circle represents a type-a vertex, while a filled circle represents a type-b vertex. (A) A path in the auxiliary graph $\widehat{G}_{\widetilde{M}}$. (B) A cycle in the auxiliary graph $\widehat{G}_{\widetilde{M}}$. The thick edges induce the schedule. (C) An induced schedule for the path in (A). The unmatched job slice vertex in the path translates to a hole. (D) An induced schedule without holes for the cycle in (B).	17
4.3	(A) An agreement graph over three jobs j, u, v . The processing times are $p_j = 1, p_u = 3$ and $p_v = 2$. (B) The corresponding auxiliary graph.	21
4.4	(A) A path in the auxiliary graph $\widehat{G}_{\widetilde{M}}$. (B) A cycle in the auxiliary graph $\widehat{G}_{\widetilde{M}}$. The thick edges between type-b vertices induce the schedule. (C) An induced schedule for the path in (A). The ghost vertex in the path translates to a hole. (D) An induced schedule for the cycle in (B). As opposed to the analog case in Fig. 4.2, the schedule corresponding to a cycle may contain holes.	23

4.5	The three types of legal cuts	25
4.6	The case analysis of the definition of the next cut. For each segment between two cuts, the induced weighted matching is depicted. The first (leftmost) column deals with cases in which the bottom cut is of type I. Note that we skip the case that the first job above a cut of type I is a unit job. The reason is that this case is identical to the case where the cut is of type II. The second and third columns deal with cases in which the bottom cut is of type II. The last column deals with cases in which the bottom cut is of type III. Ghost vertices are depicted by squares with an “H”.	25
5.1	The figure shows an example of the reduction in the proof of Lemma 5.2.1. (A) An agreement graph \overline{G} with three jobs j, u, v where $p_j = 1 + \epsilon$ for a small $\epsilon > 0$, $p_u = 1.5, p_v = 2$. (B) The agreement graph over \tilde{J} (after rounding down the jobs). In this example all three jobs are in the same bin since their processing times is between 2^0 and 2^1 . (C) A schedule produced by the approximation algorithm of m -set-cover over \tilde{J}_i . (D) The schedule over J induced by the schedule over \tilde{J} . (E) An optimal schedule over J	29

List of Tables

1.1	Approximation ratios for scheduling with conflicts. The first two algorithms have running time exponential in m while the third has running time polynomial in m	5
-----	--	---

Abstract

We consider the following problem of scheduling with conflicts (SWC): Find a minimum makespan schedule on a constant number of identical machines where conflicting jobs may not be scheduled concurrently. We present the first approximation algorithms for the case in which conflicts between jobs are modeled by general graphs both for unit jobs and jobs with arbitrary processing times.

Previous results showed that the problem is weakly NP-hard for two machines and arbitrary processing times, and strong NP-hardness is known to hold in the unit case for at least three machines. Hence, we consider approximation algorithms for general number of machines, and short jobs for two machines.

Three different approaches are considered for developing algorithms for SWC. First, SWC with unit jobs can be formulated as a set-cover problem in which the number of elements in each set is bounded by the number of machines. This gives an approximation ratio of $\mathcal{H}_m - \frac{1}{2} = O(\log m)$, where m is the number of machines. The running time of this algorithm is exponential in m . For jobs with arbitrary processing times, we extend this technique by applying scaling. Scaling increases the approximation ratio to $O(\log \frac{\max_j p_j}{\min_j p_j} \cdot \log m)$ where p_j is the processing time of a job j . Second, we analyze the greedy algorithm for arbitrary processing times. We present the first analysis of the approximation ratio of the greedy algorithm, and prove that it equals $\frac{m+1}{2}$. We also show the tightness of this analysis, even for unit jobs. Third, we focus on short jobs and two machines. For processing times in the set $\{1, 2\}$ we introduce an optimal algorithm, that is based on finding a maximum matching in an auxiliary graph. This result is related to finding a maximum bipartite b -matching, for $b = 2$. When the processing times are from the set $\{1, 2, 3\}$, a min-cost matching is used to achieve a $\frac{4}{3}$ -approximation ratio is achieved. We also show that the analysis is tight.

Chapter 1

Introduction

We consider the problem of *scheduling with conflicts* (SWC), defined as follows. There are m identical machines. The input consists of a set J of n jobs with processing times $\{p_j\}_{j \in J}$, and a conflict graph $G = (J, E)$ over the jobs. Each edge in E models a pair of conflicting jobs that may not be scheduled concurrently (on different machines). A *schedule* is an assignment of time intervals on the m machines to the jobs that satisfies the following conditions: (i) each job $j \in J$ is assigned an interval of length p_j on one machine; (ii) intervals on the same machine do not overlap; and (iii) intervals assigned to conflicting jobs do not overlap. The *makespan* of a schedule is the largest endpoint of an interval assigned to a job. We are interested in schedules that minimize the makespan.

Motivation. The problem of scheduling with conflicts models situations in which a set of resources R_j is associated with each job j . A conflict between jobs j_1 and j_2 arises if they require the same resource, namely, $R_{j_1} \cap R_{j_2} \neq \emptyset$. We consider two examples for scheduling with conflicts.

In the first example, there are two lecture halls, courses (divided into freshmen, sophomore, junior, and senior classes), and lecturers (each lecturer may teach more than one course). Two courses that belong to the same class may not be scheduled concurrently (because students of the same class must attend all these courses). In addition, two courses that are taught by the same lecturer may not be scheduled concurrently. Here, the resource of a course consists of the class of students that attend it and the lecturer.

The example second models scheduling messages in a wireless time-division multiplexed network. The network consists of m stations and n clients. A pair (i, j) specifies a message sent

from station i to client j . Two messages (i_1, j_1) and (i_2, j_2) may not be transmitted concurrently if: (i) $i_1 = i_2$, namely, the same station transmits these messages, (ii) j_1 is in the range of i_2 , namely, the transmission from i_2 disturbs reception in j_1 , or (iii) j_2 is in the range of i_1 . Figure 1.1 depicts two stations and pairs of conflicting and nonconflicting messages. In this example the resources $R_{(i,j)}$ of a message (i, j) consist of two parts: the station i and the set $R_j \triangleq \{i' : i' \neq i \ \& \ j \in \text{range}(i')\}$. A conflict between two messages (i_1, j_1) and (i_2, j_2) arises if $i_1 = i_2$ or $i_1 \in R_{j_2}$ or $i_2 \in R_{j_1}$.

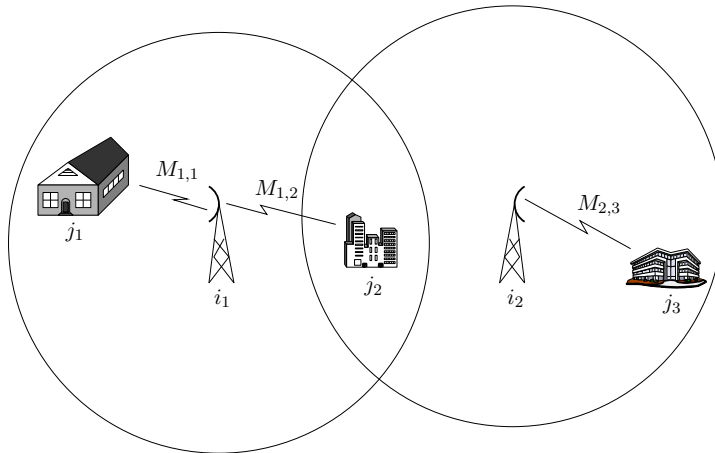


Figure 1.1: Scheduling with conflicts in a wireless network. Suppose there are three messages: (1) $M_{1,1}$ from i_1 to j_1 , (2) $M_{1,2}$ from i_1 to j_2 , and (3) $M_{2,3}$ from i_2 to j_3 . Message $M_{1,2}$ conflicts both with $M_{1,1}$ (same station) and $M_{2,3}$ (disturbance from i_2). Messages $M_{1,1}$ and $M_{2,3}$ do not conflict.

Scheduling with conflicts generally arises as resource-constrained scheduling [GG75]. In this setting there is a set of resources, each with a certain supply. Each job specifies its demand for each resource. A conflict arises between a subset of jobs if their cumulative demand of a resource exceeds its supply. In general this setting can be modeled by a conflict hypergraph. In special cases, a conflict graph suffices, e.g., if the resources are nonsharable. Baker & Coffman [BC96] present an application of this type for balancing the load in a parallel computation. Halldórsson et. al. [HKP⁺03] mention other applications in traffic intersection control, frequency assignment in cellular networks, and session management in local area networks. Bodlaender & Jansen [BJ95] present an application derived from a problem of assigning operations to processors, where the operations are given in a flow graph.

Problems involving conflict scheduling can be classified according to the following parameters: (i) The objective: minimum makespan, minimum sum of completion times (sum), maximum response time (latency). (ii) Job model: unit jobs, arbitrary processing times with/without

preemption, batch scheduling. (iii) Number of machines m : fixed or unlimited. In this paper we focus on the case characterized by the min makespan objective, nonpreemptive processing, and a fixed number of machines. We consider the cases of unit jobs, short jobs, and arbitrary processing times. One may further characterize problems by: (iv) Online/Offline model: In the online model, jobs arrive at different release times. (v) Conflict graph: general or belonging to specific classes (e.g., trees, interval graphs, etc.) We consider the offline model, and treat general graphs.

1.1 Previous results

To the best of our knowledge, SWC has been studied previously only for unit jobs in the offline model, sometimes under the name *mutual exclusion scheduling* [BC96]. An optimal algorithm for two machines and unit jobs based on matching is described in [GJ75, BC96].

By a reduction from PARTITION [GJ79], SWC is weakly NP-hard for two machines and arbitrary processing times even if the conflict graph is empty. Strong NP-hardness is known to hold in the unit case for $m \geq 3$ in general graphs [BC96] and for various special graphs, including: complements of comparability graphs for $m \geq 3$ [Lon92], interval graphs for $m \geq 4$ [BJ93], and permutation graphs for $m \geq 6$ [Jan03]. In fact, the case of unit jobs and $m = 3$ is APX-hard. Petrank [Pet94] proved that it is NP-hard to distinguish between instances of 3-Dimensional-Matching (3DM) that admit a perfect matching and instances that admit only matchings that cover a fraction of $(1 - \epsilon)$ of the elements. The same proof also implies a hardness gap for maximum packing of triangles in a graph. This, in turn, implies a hardness gap for scheduling with conflicts of unit jobs on 3 machines. For unbounded m , the unit jobs case is equivalent to the classic graph coloring problem. It is known to be highly difficult to approximate, as obtaining an approximation ratio of $n^{1-\epsilon}$ is NP-hard [FK98, Zuc06], for any $\epsilon > 0$.

SWC with unit jobs on m machines is equivalent to finding a minimum coloring of the conflict graph in which every color class contains at most m vertices, known as an *m -bounded coloring*. The correspondence between these problems is based on assigning a different color to each time unit in the schedule, thus the number of colors equals the makespan of the schedule. This case was considered by Baker and Coffman [BC96], who gave a greedy algorithm that yields a schedule whose makespan is within an additive d_k of the optimal, where $k = \lceil n/m \rceil + 1$ and d_k is the k th largest degree in the graph. However, this result does not

give a nontrivial approximation ratio if the k -th largest degree is close to n . Special graphs for which the m -bounded coloring problem (SWC with unit jobs) is polynomially solvable include: forests [BC96], split graphs and complements of interval graphs [Lon92], cographs and bipartite graphs [BJ93, HHK93], constant treewidth [KGS95], and line graphs [Alo83]. In [BC96], a constant approximation algorithm is presented for the case in which the conflict graph is obtained from a two-dimensional domain decomposition problem.

1.2 Our results

We present the first approximation algorithms for SWC for general conflict graphs. The approximation ratios in the model are summarized in Table 1.1. Since a single machine scheduler is a trivial m -approximate algorithm, the challenge is to achieve approximation ratios smaller than m . We first observe that SWC with unit jobs can be formulated as a set cover problem with sets of size at most m . This gives an approximation ratio of $\mathcal{H}_m - \frac{1}{2}$ due to a result of Duh and Fürer [DF97]. Their algorithm is polynomial only for constant values of m since the running time is exponential in m . By applying scaling, the approximation ratio for unit jobs can be extended to arbitrary processing times. Scaling incurs an increase in the approximation ratio that is logarithmic in the ratio between the maximum processing time and the minimum processing time.

We deal with the case of arbitrary processing times by analyzing the greedy algorithm whose running time is polynomial in both n and m . We prove that its approximation ratio is $\frac{m+1}{2}$, and present tight examples with unit jobs for this approximation ratio. Interestingly, the same analysis holds for the greedy algorithm in a more general setting of resource-constrained scheduling. Namely, where resources have multiple copies and each job may request a different number of copies of each resource.

We also present the following improved results for the case of short jobs on two machines: an optimal algorithm for two machines with processing times $p_j \in \{1, 2\}$ and a $4/3$ -approximation for $p_j \in \{1, 2, 3\}$. These algorithms are based on finding a maximum matching or a minimum weight perfect matching in an appropriate auxiliary graph. Interestingly, the algorithm for $p_j \in \{1, 2\}$ is related to finding a maximum bipartite b -matching for $b = 2$.

processing times p_j	#machines m	approximation ratio	technique
unit	m	$\mathcal{H}_m - \frac{1}{2}$	m -set-cover
arbitrary	m	$O(\log \frac{\max_j p_j}{\min_j p_j} \cdot \log m)$	m -set-cover + scaling
arbitrary	m	$\frac{m+1}{2}$	greedy algorithm
$\{1, 2\}$	2	1 (optimal)	max matching
$\{1, 2, 3\}$	2	$4/3$	min cost matching

Table 1.1: Approximation ratios for scheduling with conflicts. The first two algorithms have running time exponential in m while the third has running time polynomial in m .

1.3 Other Related Work

A special type of resource-constrained problem, in fact a subproblem of SWC, is the scheduling of multiprocessor tasks with dedicated processors (see [Dro96] for a dated review). Each job is given with a specified set of processors that it requires the exclusive use of during its execution. Here, the processors are the resource, and the number of resources is m while the number of “machines” can be viewed to be unlimited. For m fixed, there exists a linear time fully-polynomial approximation scheme by Jansen and Porkolab [JP02]. Note that in this case the conflict graph is of constant size.

By varying the scheduling parameters, a large body of research opens up. In the case of unbounded number of machines, the focus has been on special graph classes, due to the aforementioned hardness for graph coloring. In our case of nonpreemptive makespan scheduling, two classic problems are that of *dynamic storage allocation* [BKK⁺04] (corresponding to interval graphs) and *file transfer problem* [CGJL85] (corresponding to line graphs). If we additionally change to unit jobs or preemptive scheduling, we get the classical graph coloring or multicoloring problems, of which there is a bottomless literature. Instead, if we modify the objective function, then conflict (nonpreemptive and preemptive) scheduling with the sum of completion times measure was introduced in [BHK⁺00] and studied in several recent works.

Bodlaender et. al. [BJW94] considered a complementary problem called scheduling incompatible jobs (CIJ). In CIJ, incompatible jobs cannot be processed by the same machine. They gave an approximation algorithm for CIJ with arbitrary processing times whose performance is good in the case that the graph can be colored with few colors $k < m$. The ratio obtained approaches 2 as m/k increases, and is at most $(m+1)/2$ for $k = m - 1$. CIJ was shown to be hard to approximate within any factor less than 2, for any $m \geq 3$. They also gave approximations

for the special classes of bipartite, bounded-treewidth, and complete graphs.

Somewhat related are papers on bin-packing with conflicts, where in our terms the jobs are nonunit, the makespan is restricted, and the goal is to minimize the number of machines [Jan99, EL06].

In an online setting jobs arrive over time, and the algorithm does not know the arrival time. Irani & Leung [IL03] studied a version involving unit jobs, unbounded number of machines, with the objective of minimizing the maximum response time (i.e. job completion time minus its release time). Further, they restricted their attention to bipartite and interval graphs. In continuation to the work presented in this thesis [EHKR] an online version of the SWC problem is studied.

Chapter 2

Preliminaries

2.1 The model

In a scheduling problem with conflicts on m machines, the input consists of (i) a set J of n jobs, (ii) a (simple) conflict graph $G = (J, E)$ over the jobs, and (iii) an integral processing time p_j for each job $j \in J$. Adjacent jobs in G are conflicting and cannot be scheduled concurrently on different machines. Each job j consists of p_j *job slices*, each of unit length. Time is discrete and proceeds in rounds. We use the notation $[t_1, t_2]$ for $t_1 \leq t_2$ to denote the set $\{t_1, t_1 + 1, \dots, t_2\}$. We refer to a pair $\langle i, t \rangle$, where i is a machine and t is a round, as a *machine slot*. A set of slots $\{\langle i, t \rangle : t \in [t_1, t_2]\}$ is called an *interval* of slots. Two slots $\langle i_1, t \rangle$ and $\langle i_2, t \rangle$ with the same round component are called *concurrent*.

A *schedule* T is a one-to-one assignment from job slices to slots that satisfies the following two properties: (i) *Nonpreemptive*: the slices of a job are assigned to an interval of slots on the same machine. (ii) *Nonconflicting*: job slices assigned to concurrent slots do not belong to conflicting jobs. Note that since the schedule is one-to-one and preemption is prohibited, the number of slots in the interval of slots assigned to job j equals p_j .

The *makespan* of a schedule T is the last round in which a slot is assigned a job slice. We denote the makespan of a schedule T by $\text{makespan}(T)$. The goal in this paper is to find a schedule with a minimum (or approximately minimum) makespan.

We concentrate on *deterministic scheduling algorithms* in the offline model .

2.2 Additional terminology

The complement of the conflict graph G is the *agreement graph* $\overline{G} = (J, \overline{E})$. Namely, the agreement graph is a simple graph in which $(j_1, j_2) \in \overline{E}$ if and only if $(j_1, j_2) \notin E$. Since independent sets in the conflict graph correspond to cliques in the agreement graph, jobs that form a clique of size m in the agreement graph \overline{G} can be scheduled concurrently.

A slot that is not assigned a job slice is called *hole*. The set of holes in a schedule T is denoted by $holes(T)$, and the number of holes is denoted by $|holes(T)|$. A round in which all slots are holes is called a *vacant round*.

Let $p(J') \triangleq \sum_{j \in J'} p_j$. In particular, the total amount of work that needs to be processed is $p(J)$.

For a given input, let OPT denote a schedule with minimum makespan and let $|OPT|$ denote its makespan.

Chapter 3

m Machines & Arbitrary Processing Times

In this chapter we analyze the makespan of the greedy mutual exclusion (GME) algorithm in the case of m machines and jobs of arbitrary (integral) processing times. We note that the algorithm and its analysis can be extended to non-integral processing times. We prove that GME obtains an $(m + 1)/2$ -approximation of the makespan, and show this ratio to be tight for this algorithm.

A listing of the GME algorithm appears below. Note that, in the description of the algorithm, after a hole is scheduled in a machine's slot, then the machine is not considered vacant during this slot anymore.

GME($G = (J, E), m$) : Greedy Mutual Exclusion schedules the jobs J with conflict graph G on m machines.

```
1: while  $J \neq \emptyset$  do
2:   Let  $t$  denote the first round that contains a vacant slot.
3:   if exists a job  $j \in J$  that agrees with all the jobs already scheduled in round  $t$  then
4:     Pick a machine that has a vacant slot in round  $t$  and schedule  $j$  on that machine during
       rounds  $[t, t + p_j - 1]$ .
5:      $J \leftarrow J \setminus \{j\}$ 
6:   else
7:     Schedule a hole in round  $t$  on each machine that has a vacant slot in round  $t$ .
8:   end if
9: end while
```

For simplicity we described GME round by round, so that the running time of the algorithm, as described, is pseudo-polynomial. The algorithm can quite easily be modified to run in (strictly) polynomial time by working in phases as follows. In each phase we consider a machine with minimal load (that is, the smallest number of occupied slots). If we can schedule a job on that machine, we do so. Otherwise, we schedule a *block (sequence) of holes* on this machine (and all other machines that have the same load), so that their load now equals the second largest load. It follows that a job is scheduled every $m' \leq m$ phases.

3.1 Analysis of GME's makespan

We now prove the following lemma.

Lemma 3.1.1 *GME is an $(\frac{m+1}{2})$ -approximation algorithm for the makespan.*

Proof: Fix an input and consider the schedule that is computed by an execution of GME. Let $slices(t)$ denote the set of job slices that are scheduled in round t by GME. Let A_k denote the set of rounds in which k jobs are scheduled (namely, $t \in A_k$ iff $|slices(t)| = k$). Let $A_{\geq 2} \triangleq \bigcup_{i=2}^m A_i$ (note that $|A_{\geq 2}| = \sum_{i=2}^m |A_i|$ since the sets A_i are disjoint).

The proof is based on the following two equations:

$$|A_1| \leq |OPT| \tag{3.1}$$

$$|A_{\geq 2}| \leq \frac{1}{2}(p(J) - |A_1|) \tag{3.2}$$

Equation (3.1) holds because the set of jobs that have slices in $\bigcup_{t \in A_1} slices(t)$ form an independent set in the agreement graph \overline{G} . Indeed, suppose j_1 and j_2 contain slices that are scheduled in rounds $t_1, t_2 \in A_1$, respectively, where $t_1 < t_2$. If j_1 and j_2 are not conflicting jobs, then j_2 would have been scheduled by GME in round t_1 together with j_1 . In this case, round t_1 would not be in A_1 .

Equation (3.2) holds because GME schedules $p(J) - |A_1|$ job slices during the rounds $A_2 \cup \dots \cup A_m$, at least two per round. Observe that the sum $p(J) = \sum_j p_j$ of processing times satisfies

$$p(J) = m \cdot |OPT| - |holes(OPT)| \tag{3.3}$$

GME's makespan equals $|A_1| + |A_{\geq 2}|$, which we bound as follows:

$$\begin{aligned}
|A_1| + |A_{\geq 2}| &\leq \frac{1}{2} \cdot (p(J) + |A_1|) && \text{(by Eq. 3.2)} \\
&\leq \left(\frac{m+1}{2}\right) \cdot |OPT| - \frac{|\text{holes}(OPT)|}{2} && \text{(by Eq. 3.1 and Eq. 3.3)} \\
&\leq \left(\frac{m+1}{2}\right) \cdot |OPT|,
\end{aligned}$$

and the lemma follows. ■

The proof of Lemma 3.1.1 implies that holes in the optimal schedule (i.e., $|\text{holes}(OPT)| > 0$) reduce the approximation ratio of GME. For example, if $|\text{holes}(OPT)| \geq \frac{m}{2} \cdot |OPT|$, then the approximation ratio of GME is reduced to $\frac{m}{4} + \frac{1}{2}$.

3.2 Tightness of the approximation of the makespan

We present an example with m^2 unit jobs over m machines with the following property. All optimal schedules have a makespan m and lack holes. However, GME might compute a “bad” schedule in which only two machines are utilized. Moreover, in this bad schedule there are m jobs that GME schedules alone. This implies that the ratio of $(m+1)/2$ is indeed tight.

The set of jobs is $J = \{(i, \ell) \mid (i, \ell) \in [0, \dots, m-1] \times [0, \dots, m-1]\}$. The edges of the agreement graph $\overline{G} = (J, \overline{E})$ form m cliques of size m , where the i th clique is over the set of vertices $\{i\} \times [0, \dots, m-1]$. Suppose that m is even. We add “matching” edges $((2i, \ell), (2i+1, \ell))$, for each $i \in [0, \dots, m/2-2]$ and each $\ell \in [0, \dots, m-2]$. Note that, all the vertices, except the “last” one, are matched in each clique. Figure 3.1 (A) depicts the agreement graph.

An optimal schedule can schedule each clique in a single round, thus completing the schedule in m rounds. GME, on the other hand, might select pairs of jobs that are connected by matching edges. Each such pair is a maximal clique in the agreement graph, namely, no other job can be scheduled concurrently. After scheduling $m(m-1)/2$ pairs of matching jobs, GME is left with m independent jobs in \overline{G} . These jobs are then scheduled as singletons, as claimed above.

A similar construction exists for odd values of m . Here, each clique is partitioned into 3 parts: (a) the left part $\{i\} \times [0, \dots, (m-1)/2-1]$, (b) the right part $\{i\} \times [(m-1)/2, m-2]$, and (c) the last vertex $(i, m-1)$. Note that the left and right parts each contain $(m-1)/2$ vertices.

Now, we add edges that form a perfect matching from the right part of clique i to the left part of clique $i + 1 \pmod{m}$. Figure 3.1 (B) depicts the agreement graph in this construction. A similar argument shows that the greedy algorithm might compute a schedule whose makespan is $m + m(m - 1)/2$.

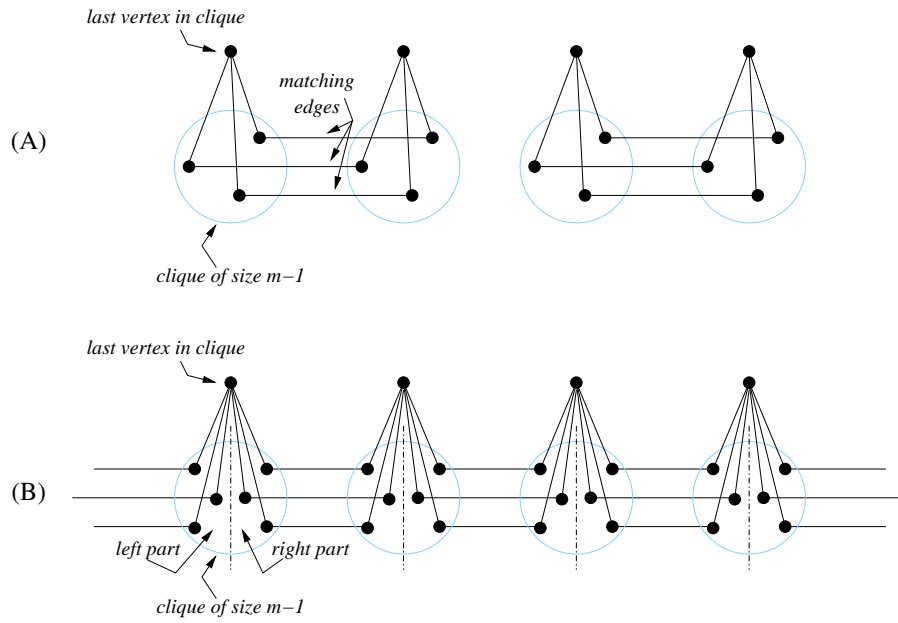


Figure 3.1: Tight constructions for the greedy algorithm. (A) A construction for the case that m is even. (B) A construction for odd values of m . The m cliques are arranged in a cycle.

Chapter 4

Two Machines & $p_j \in \{1, 2, 3\}$

In this chapter we present two results for scheduling on two machines. These results improve on the upper bound of $\frac{m+1}{2} = \frac{3}{2}$ that GME achieves on two machines (and arbitrary job sizes). The first result is an optimal algorithm for the case that the processing times satisfy $p_j \in \{1, 2\}$. The second result is a $4/3$ -approximation algorithm for the case that $p_j \in \{1, 2, 3\}$. These algorithms rely on computing a maximum matching or a minimum weight perfect matching in an appropriate auxiliary graph. We first recall what is known for unit jobs and two machines.

4.1 Unit jobs

Garey and Johnson presented the following optimal algorithm for the case of unit jobs [GJ75] (see also [BC96]). Compute a maximum matching M in the agreement graph \overline{G} . Matched pairs are scheduled concurrently, and the unmatched jobs are scheduled as singletons. The optimality of this algorithm is based on the following observation.

Observation 4.1.1 *In the case of unit jobs, every matching M in the agreement graph, induces a schedule with makespan of $|J| - |M|$. Every schedule T without vacant rounds induces a matching M_T in \overline{G} , such that $|M_T| = |J| - \text{makespan}(T)$.*

4.2 An Optimal Algorithm for Processing Times $p_j \in \{1, 2\}$

4.2.1 Intuition

Consider a schedule on two machines and $p_j \in \{1, 2\}$. Such a schedule corresponds to a matching between job slices (of unit size, as defined in the preliminaries). Namely, each pair of

job slices that are scheduled concurrently, are matched, and a job slice that has a hole opposite it is left unmatched. Note that for all pairs of matched job slices, the corresponding jobs have an edge between them in the agreement graph. Given this matching, the resulting makespan is exactly the sum of processing times minus the size of the matching.

In view of the above description, the high level idea of the algorithm is to create an auxiliary graph whose vertices correspond to job slices. We then find a maximum matching in this graph and transform the matching into a schedule. Care must be taken when defining the graph that the matching found indeed induces a feasible schedule. In particular, it must be possible to schedule different slices of the same job consecutively. Details follow.

4.2.2 Auxiliary graph

To simplify notation, we refer to unit jobs in this section with the letter j . Jobs whose processing time equals 2 are called *double size jobs* (in short, double jobs) and are named with the letters u and v .

We define an auxiliary graph $\tilde{G} = (\tilde{J}, \tilde{E})$ (see Fig. 4.1). The vertex set \tilde{J} contains one vertex for each unit job $j \in J$, and two vertices for each double job $v \in J$ as follows. Each double job v is split into two slices: a *type-a* vertex v_a and a *type-b* vertex v_b . The edge set \tilde{E} consists of *agreement edges* between slices. Loosely speaking, agreement edges are induced by the agreement graph subject to the constraint that there are no agreement edges between a type-a vertex and a type-b vertex. Formally,

$$\tilde{E}_A \triangleq \{(j, v_\sigma) \mid \sigma \in \{a, b\}, (j, v) \in \overline{E}\} \cup \{(v_\sigma, u_\sigma) \mid \sigma \in \{a, b\}, (v, u) \in \overline{E}\}.$$

We refer to edges between type-a vertices (v_a, u_a) as type-a edges, and to edges between type-b vertices (v_b, u_b) as type-b edges.

Given the auxiliary graph \tilde{G} and a matching M , let \hat{G}_M denote the (multi) graph obtained from \tilde{G} and M as follows (see Fig. 4.2). Consider the subgraph of \tilde{G} induced by M . For each double job $v \in \tilde{J}$, coalesce the vertices v_a and v_b in \tilde{J} into a single vertex. Namely, in $\hat{G}_M = (\hat{J}, \hat{E})$, for each double job $v \in \tilde{J}$, coalesce the vertices v_a and v_b in \tilde{J} into a single vertex \hat{v} in \hat{J} . The vertices that correspond to unit jobs remain as in \tilde{J} . For each edge in the matching M , there is an edge in \hat{E} . If an edge in M is incident to a vertex v_a or v_b in \tilde{J} , then the corresponding edge in \hat{E} is incident to the vertex \hat{v} in \hat{J} . Note that each vertex in \hat{G}_M is incident to at most two edges.

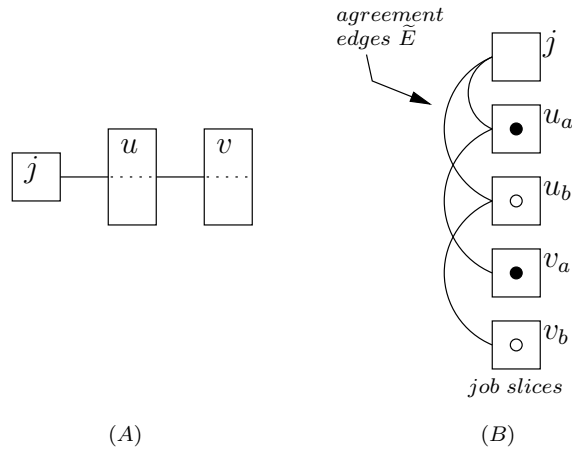


Figure 4.1: (A) An agreement graph over three jobs j, u, v . The processing times are $p_j = 1$ and $p_u = p_v = 2$. (B) The corresponding auxiliary graph. Job slices of type-a are depicted using a filled circle while job slices of type-b are depicted using an unfilled circle. Unit jobs lack a circle.

Claim 4.2.1 For any matching M in \tilde{G} , the graph \hat{G}_M is bipartite.

Proof: Since every vertex in \hat{G}_M has a degree of at most 2, the graph is a union of disjoint paths and cycles. Each path is, clearly, bipartite. Consider a cycle τ in \hat{G}_M (see Fig. 4.2 (B)). Since all vertices in the cycle τ are incident to two edges in \hat{G}_M , it follows that each vertex in τ corresponds to a double job. Each edge (\hat{u}, \hat{v}) in τ corresponds either to an edge (u_a, v_a) in the matching M or to an edge (u_b, v_b) in M (since edges in \tilde{E} do not connect slices of different types). Each coalesced vertex \hat{v} contains one type-a slice and one type-b slice. It follows that the edges of τ alternate between type-a edges and type-b edges. Hence, the cycle has even length. ■

4.2.3 Scheduling algorithm

Scheduling algorithm.

1. Compute a maximum matching \tilde{M} in \tilde{G} .
2. Partition $\hat{G}_{\tilde{M}}$ into paths and cycles.
3. For each path or cycle τ in $\hat{G}_{\tilde{M}}$, obtain a schedule T_τ of corresponding jobs as described in the proof of Claim 4.2.2 below.
4. Return the concatenation of the schedules $\{T_\tau\}$.

Let τ be a path or a cycle in \widehat{G}_M . Denote by J_τ the set of jobs that correspond to vertices in τ , and by M_τ the set of edges in τ .

Claim 4.2.2 *For any given matching M in \widetilde{G} , consider the auxiliary graph \widehat{G}_M . Each path or cycle τ in \widehat{G}_M induces a schedule T_τ of the jobs corresponding to the vertices in τ , such that $\text{makespan}(T_\tau) = p(J_\tau) - |M_\tau|$.*

Proof: First consider a path τ in \widehat{G}_M . Note that unit job vertices are incident to at most one edge in \widehat{G}_M . Hence, they may appear only as end vertices in the path τ . Interior vertices in τ correspond to double jobs. This implies that the path induces a schedule in which jobs alternate between the machines as depicted in Fig. 4.2 (C). The only holes, if any, in this schedule are in the first and last round. A hole appears in the schedule if the path begins or ends with a double job of degree at most 1, or a unit job of degree 0. Hence, the makespan of the schedule is the number of job slices in τ minus the number of pairs of matched slices (that are scheduled concurrently). Therefore, $\text{makespan}(T_\tau) = p(J_\tau) - |M_\tau|$ as claimed.

Next consider a cycle τ in \widehat{G}_M . By Claim 4.2.1, the cycle τ is of even length. Since there is an edge (\hat{u}, \hat{v}) in \widehat{G}_M only if there is an edge between the corresponding double jobs u and v in the agreement graph \overline{G} , the type-b edges form a perfect matching in \overline{G} between the jobs that correspond to vertices on the cycle τ (see Fig. 4.2 (D)). And the claim follows. \blacksquare

4.2.4 Correctness

Optimality of the algorithm is derived from Lemmas 4.2.3 and 4.2.4, stated below.

Lemma 4.2.3 *Every matching M in the auxiliary graph \widetilde{G} induces a feasible schedule T_M , such that $\text{makespan}(T_M) = p(J) - |M|$.*

Lemma 4.2.3 follows immediately from Claim 4.2.2.

Lemma 4.2.4 *Every schedule T induces a matching M_T in the auxiliary graph \widetilde{G} with $|M_T| \geq p(J) - \text{makespan}(T)$.*

Equality holds in Lemma 4.2.4 if there are no vacant rounds in T .

Proof: We consider the rounds in T starting from the first round. We shall map each round with two concurrent job slices to an edge in \widetilde{G} , so that the edges form a matching. The main

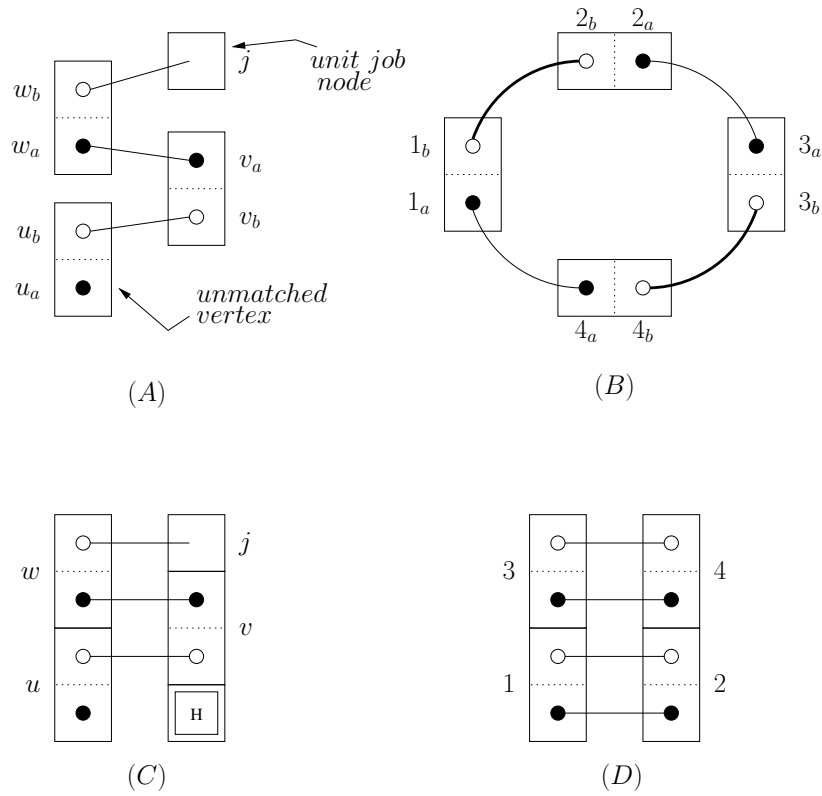


Figure 4.2: An unfilled circle represents a type-a vertex, while a filled circle represents a type-b vertex. (A) A path in the auxiliary graph $\hat{G}_{\tilde{M}}$. (B) A cycle in the auxiliary graph $\hat{G}_{\tilde{M}}$. The thick edges induce the schedule. (C) An induced schedule for the path in (A). The unmatched job slice vertex in the path translates to a hole. (D) An induced schedule without holes for the cycle in (B).

issue will be in selecting the types (a or b) for job slices of double jobs. There are three types of rounds with two concurrent job slices scheduled in them:

1. Unit jobs j and k are scheduled concurrently in the round. In this case there must be an edge in \tilde{G} between the corresponding vertices, and we select this edge.
2. Unit job j is scheduled concurrently with a job slice of a double job v . If this is the first slice of job v , or if this is the second slice of job v and the first one was scheduled opposite a hole, then we select the edge $(j, v_a) \in \tilde{E}$. Otherwise (it is the second slice and the first one was not scheduled opposite a hole), the previous round was mapped to an edge (x, v_a) (or (x, v_b)), and we select (j, v_b) (or, respectively, (j, v_a)).
3. A slice of a double job v is scheduled concurrently with a slice of another double job u . If they are both the first slice of their respective jobs, then we select (v_a, u_a) . If they are both the second slice of their respective jobs, then we select (v_b, u_b) . Otherwise, one of them is a second slice of its job, if its first slice was scheduled opposite to a hole we select (v_a, u_a) . Otherwise, the type selected for its first slice determines its type, so we select either (v_a, u_a) or (v_b, u_b) accordingly.

■

Theorem 1 *The algorithm for scheduling jobs with processing times $p_j \in \{1, 2\}$ computes a schedule with minimum makespan.*

Proof: Let \tilde{M} denote the maximum matching in the auxiliary graph found by the scheduling algorithm. Let T^* denote a schedule with minimum makespan, and let M_{T^*} denote the matching induced by T^* as defined in Lemma 4.2.4.

$$\begin{aligned}
 \text{makespan}(T_{\tilde{M}}) &= p(J) - |\tilde{M}| && \text{(by Lemma 4.2.3)} \\
 &\leq p(J) - |M_{T^*}| && (\tilde{M} \text{ is maximum}) \\
 &\leq \text{makespan}(T^*) && \text{(by Lemma 4.2.4).}
 \end{aligned}$$

■

4.2.5 On the relation to b -matching

Loosely speaking, a b -matching is a multi-set of edges F , where the degree of each vertex v with respect to F is at most b_v (an edge can be chosen more than once). The algorithm we showed above solves the problem of finding a maximum bipartite b -matching where $b_j = p_j$. This algorithm works only when $p_j \in \{1, 2\}$.

4.3 $\frac{4}{3}$ -Approximation for Processing Times $p_j \in \{1, 2, 3\}$

4.3.1 Intuition

First, we look at the case of two machines and $p_j \in \{1, 2\}$ a little differently, and then we extend this to $p_j \in \{1, 2, 3\}$.

Consider a schedule on two machines and $p_j \in \{1, 2\}$. It will be convenient to view the holes in this schedule (if any) as *ghost* unit jobs. Such a schedule induces a matching between job slices (of unit size, as defined in the preliminaries), where these slices include the ghost unit jobs. For all pairs of job slices of “real” jobs that are matched, the corresponding jobs have an edge between them in the agreement graph. If we assign weight 1 to each edge between a ghost unit job and a real job slice (and assign zero weight to all other edges), then the weight of this matching exactly equals the number of holes in the schedule.

Given the above description, the high level idea of the algorithm for $p_j \in \{1, 2\}$ is to create an auxiliary weighted graph whose vertices correspond to job slices of real jobs and ghost jobs. We then find a minimum weight perfect matching in the auxiliary graph and transform the matching into a schedule. The approximation algorithm for processing times $p_j \in \{1, 2, 3\}$ is based on the idea of building an auxiliary graph with edge weights in $\{0, 1, 2\}$. Finding a minimum weight perfect matching in the auxiliary graph induces a schedule with $4/3$ -ratio. Details follow.

We refer to jobs j with $p_j = 3$ as *triple* size jobs (in short, triple jobs). We shall need the following notation. Recall that a hole is a slot that is not assigned a job slice, and that $holes(T)$ denotes the set of holes in the schedule T . For a matching M in the auxiliary graph, let $w(M) = \sum_{e \in M} w(e)$ denote the weight of the edges in the matching, where $w(e)$ denotes the weight of an edge.

4.3.2 Auxiliary graph

To simplify notation, we refer to unit jobs in this section with the letter j , whereas v and u are used to denote both double jobs and triple jobs.

We define an auxiliary graph $\tilde{G} = (\tilde{J}, \tilde{E})$ with edge weights $w(e)$ for each edge $e \in \tilde{E}$ (see Fig. 4.3). The vertex set \tilde{J} contains two types of vertices: job slices and their ghosts as follows. A unit job $j \in J$ induces two vertices in \tilde{J} : j and j' . A double job $v \in J$ induces four vertices in \tilde{J} as follows. First it is split into two slices: a type-a vertex v_a and a type-b vertex v_b . In addition, each slice has a corresponding ghost denoted by v'_a and v'_b respectively. Similarly, a triple job $v \in J$, induces four vertices. We split it into two *job pieces*: a type-a vertex v_a which is a job slice, and a type-b vertex v_b which is two slices glued together. Again, each piece has a corresponding ghost denoted by v'_a and v'_b respectively. In the following we refer to unit jobs, job slices and job pieces as job pieces. The edge set \tilde{E} consists of three types of edges: (i) Parallel edges \tilde{E}_P between a job piece and its ghost. The weight of a parallel edge equals the size of the piece (i.e. 2 for a type-b vertex of a triple job, and 1 otherwise). (ii) Agreement edges \tilde{E}_A between pieces. These edges are an extension of the agreement edges in the auxiliary graph in Section 4.2 to triple jobs as well. Formally,

$$\tilde{E}_A \triangleq \{(j, v_\sigma) \mid \sigma \in \{a, b\}, (j, v) \in \overline{E}\} \cup \{(v_\sigma, u_\sigma) \mid \sigma \in \{a, b\}, (v, u) \in \overline{E}\}.$$

Agreement edges that connect a type-b vertex of a triple job to a job slice or a unit job, have weight 1. All other agreement edges have zero weight. (iii) Ghost edges \tilde{E}_H between ghost vertices. We add an edge between each pair of ghost vertices. All ghost edges have zero weight. Finally, $\tilde{E} \triangleq \tilde{E}_P \cup \tilde{E}_A \cup \tilde{E}_H$.

The intuition behind the definition of edge weights is that a ghost plays the role of a hole or two holes. An edge between a piece and its ghost signifies a hole or two holes in the schedule. Therefore, the weight of such an edge is 1 or 2, respectively.

4.3.3 Scheduling algorithm

The parallel edges \tilde{E}_P in the auxiliary graph \tilde{G} form a perfect matching. Hence, \tilde{G} contains a perfect matching. The schedule induced by \tilde{E}_P simply assigns all the jobs on a single machine, and the makespan equals the weight of the matching (i.e., the sum of the job processing times). More generally, as we shall see, every perfect matching induces a schedule in which the number of holes equals the weight of the matching.

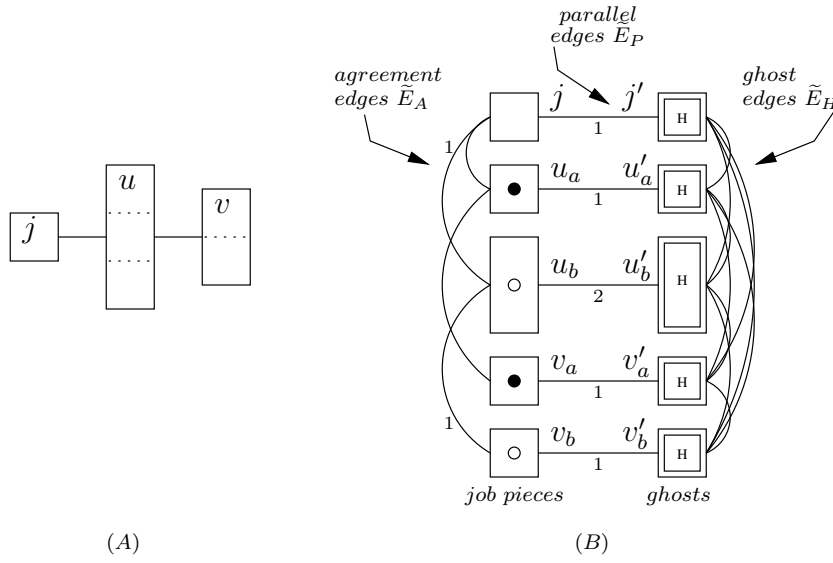


Figure 4.3: (A) An agreement graph over three jobs j, u, v . The processing times are $p_j = 1$, $p_u = 3$ and $p_v = 2$. (B) The corresponding auxiliary graph.

The algorithm proceeds as follows:

1. Compute a minimum weight perfect matching \tilde{M} in \tilde{G} (see [Edm65, Gab90]).
2. Given the auxiliary graph \tilde{G} and a matching M , let \hat{G}_M denote the graph obtained from \tilde{G} and M as follows: (i) Consider the subgraph of \tilde{G} induced by M . (ii) Delete pairs of ghosts that are matched by edges in M . (iii) For each double or triple job $v \in J$, coalesce the vertices v_a and v_b in \tilde{J} into a single vertex in an analogous manner as done for the case $p_j \in \{1, 2\}$.
3. Partition the edges of $\hat{G}_{\tilde{M}}$ into paths and cycles (since \tilde{M} is a matching, each vertex in $\hat{G}_{\tilde{M}}$ has at most two neighbors).
4. For each path or cycle τ in $\hat{G}_{\tilde{M}}$, obtain a schedule T_τ of corresponding jobs as described in the proof of Claim 4.3.1 below.
5. Return the concatenation of the schedules $\{T_\tau\}$.

Claim 4.3.1 *For any given perfect matching M in \tilde{G} , consider the auxiliary graph \hat{G}_M . Each path or cycle τ in \hat{G}_M induces a schedule T_τ of the jobs corresponding to the vertices in τ . The number of holes in the schedule T_τ equals the weight of the edges in τ .*

Proof: First consider a path τ in \widehat{G}_M . Note that unit job vertices and ghost vertices are incident to exactly one edge in \widehat{G}_M . Hence, they may appear only as end vertices in the path τ . Interior vertices in τ correspond to double or triple jobs. This implies that the path induces a schedule in which jobs alternate between the machines as depicted in Fig. 4.4 (C). The holes, if any, in this schedule that are induced by ghosts can only be in the first and last round. A hole appears in the schedule if the path begins or ends with a ghost, that is, the first and/or last edge correspond to a parallel edge. Holes are also introduced by agreement edges with positive weights (when a type-b vertex of a triple job is matched to a slice or a unit job).

Next consider a cycle τ in \widehat{G}_M (see Fig. 4.4 (B)). Since all vertices in the cycle τ are incident to two edges in \widehat{G}_M , it follows that each vertex in τ corresponds to a double job or a triple job. Similarly to Claim 4.2.1, τ is an even cycle with alternating edges between type-a and type-b edges. Each cycle τ in \widehat{G}_M induces a schedule by considering the type-b edges in τ , and adding a hole where the b-type edge has weight 1, see Fig. 4.4 (D). Note that type-a edges have zero weight, while type-b edges have weight 0 or 1, depending on their end-points. This is why type-b edges in the cycle are used for constructing the schedule. ■

4.3.4 Correctness

We use Lemmas 4.3.2 and 4.3.3 to prove Theorem 2, namely, the $4/3$ -approximation ratio.

Let M denote a perfect matching in the auxiliary graph \widetilde{G} . By Claim 4.3.1, each path or cycle in \widehat{G}_M induces a schedule. By concatenating all these schedules we obtain the following Lemma.

Lemma 4.3.2 *Every perfect matching M induces a feasible schedule T_M such that $w(M) = |\text{holes}(T_M)|$.*

Recall that $p(J) \triangleq \sum_{j \in J} p_j$.

Lemma 4.3.3 *Every schedule T induces a perfect matching M_T in \widetilde{G} such that*

$$w(M_T) \leq \frac{1}{3} \cdot p(J) + \frac{4}{3} \cdot |\text{holes}(T)|.$$

Proof: We first assume that $\text{holes}(T) = \emptyset$ and prove that $w(M_T) \leq \frac{1}{3}p(J)$.

We partition the schedule into disjoint segments, assign a matching to each segment, and show a ratio of at most 1:3 between the weight of the matching corresponding to a segment

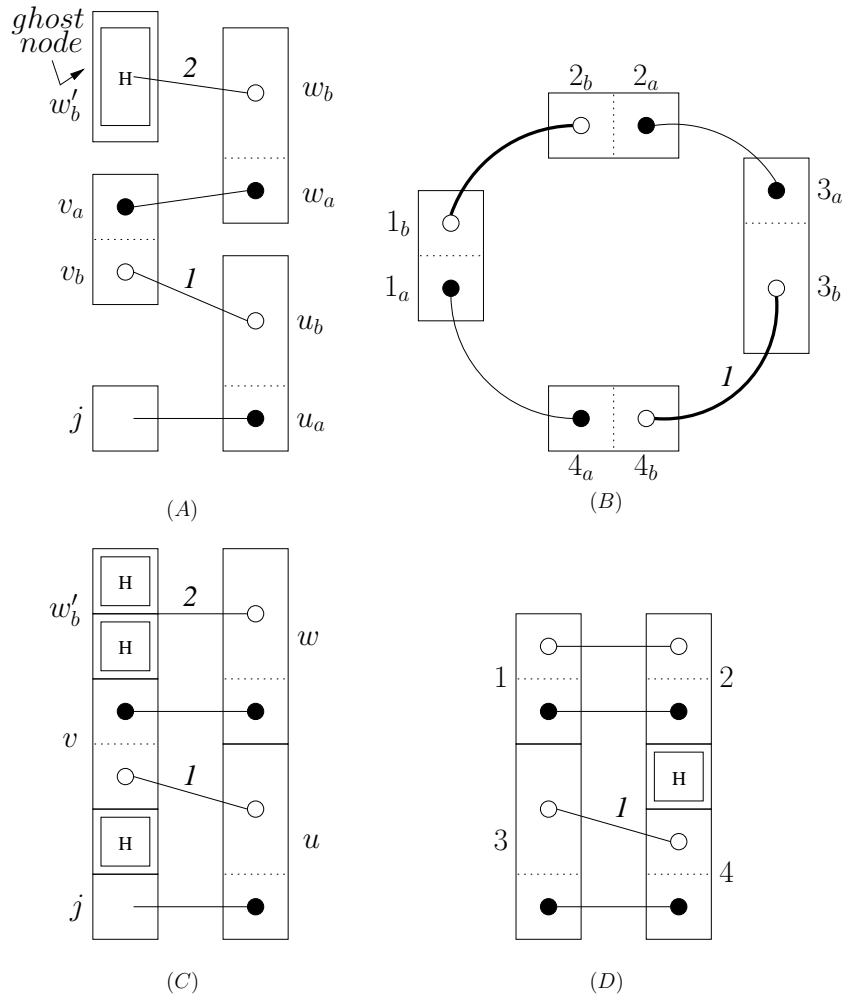


Figure 4.4: (A) A path in the auxiliary graph \widehat{G}_M . (B) A cycle in the auxiliary graph \widehat{G}_M . The thick edges between type-b vertices induce the schedule. (C) An induced schedule for the path in (A). The ghost vertex in the path translates to a hole. (D) An induced schedule for the cycle in (B). As opposed to the analog case in Fig. 4.2, the schedule corresponding to a cycle may contain holes.

and the number of job slices in the segment. Before we define the segments, types are assigned to slices and pieces of jobs as in the proof of Lemma 4.2.4. “Cuts” are used for partitioning the schedule. A pair $\langle t_1, t_2 \rangle$ specifies a cut as follows: each t_i partitions the slots on machine i to slots before round t_i and slots starting from round t_i . There are three kinds of *legal cuts* (depicted in Fig. 4.5). (I) Cuts in which $t_1 = t_2 = t$, namely, the schedule is cut in the same round on both machines. In addition, we require that no job be split by the cut, namely, no job is scheduled both to round $t - 1$ and to round t . (II) Cuts in which $t_1 = t_2 = t$. Here, we allow a job v to be split by the cut provided that the processing of v ends in round t and the slice of v in round t is a type-a slice. (III) Cuts in which $|t_1 - t_2| = 1$. Let i denote the machine that is cut first by the cut (namely, $i = 1$ if $t_1 < t_2$, and $i = 2$ if $t_2 < t_1$). In this case we require that a triple job v be scheduled during rounds $t_i, t_i + 1, t_i + 2$ on machine i .

The first cut we take is a cut of type (I) for $t = 1$. The remaining cuts are defined by induction. In the induction step, we have a cut cut_i and wish to define the next cut cut_{i+1} . An exhaustive case analysis for the induction step is depicted in Fig. 4.6. The case analysis enumerates all the possible combinations of types for cut_i and cut_{i+1} by considering all the combinations of job processing times in the 3 rounds after cut_i .

Once the cuts are defined, a matching M_i is assigned to each segment of the schedule between cut_i and cut_{i+1} . The matching M_i is incident to all jobs slices and job pieces in the segment between cut_i and cut_{i+1} . The matching M_i , for all cases, is also depicted in Fig. 4.6. The weight $w(M_i)$ is in the set $\{0, 1, 2\}$. Note that whenever the matching M_i has weight 1, then there are at least 3 job slices in the segment. Whenever the matching M_i has weight 2, then there are at least 6 job slices in the segment. Hence, a ratio of at least $1/3$ is kept between $w(M_i)$ and the number of slices in the corresponding segment.

The union of the matchings $\{M_i\}$ constitutes a matching that is incident to all job slices and job pieces in the auxiliary graph. This union of matchings can be augmented to a perfect matching in the auxiliary graph by adding zero weight edges between ghosts. The weight of the resulting perfect matching is at most $\frac{1}{3} \cdot p(J)$, as required.

To complete the proof, we consider that case that $holes(T) \neq \emptyset$. We reduce this case to the case of no holes by adding dummy unit jobs that are scheduled in the holes. Let M_{dummy} denote the perfect matching corresponding to the schedule after holes are replaced by dummy unit jobs. By the proof above $w(M_{dummy}) \leq \frac{1}{3} \cdot (p(J) + |holes(T)|)$.

We now remove these dummy unit jobs and analyze the effect on $w(M_T)$. Each hole in-

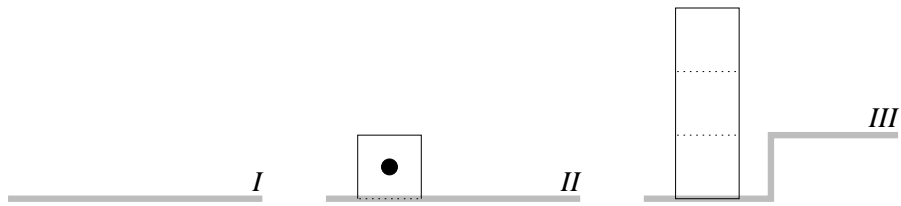


Figure 4.5: The three types of legal cuts

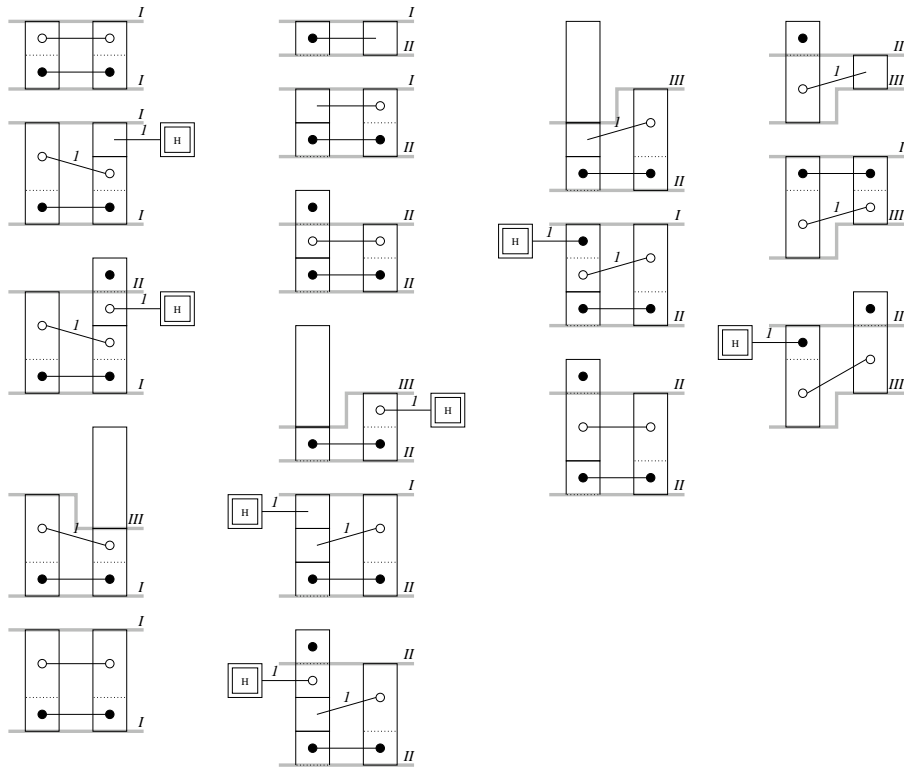


Figure 4.6: The case analysis of the definition of the next cut. For each segment between two cuts, the induced weighted matching is depicted. The first (leftmost) column deals with cases in which the bottom cut is of type I. Note that we skip the case that the first job above a cut of type I is a unit job. The reason is that this case is identical to the case where the cut is of type II. The second and third columns deal with cases in which the bottom cut is of type II. The last column deals with cases in which the bottom cut is of type III. Ghost vertices are depicted by squares with an “H”.

creases the weight of the matching by at most one. (It is possible that a dummy unit job is matched to a ghost, in which case we will drop the edge between the dummy and ghost). Therefore, $w(M_T) \leq w(M_{dummy}) + |\text{holes}(T)| \leq \frac{1}{3} \cdot p(J) + \frac{4}{3} \cdot |\text{holes}(T)|$, as required. ■

Theorem 2 *The algorithm for scheduling jobs with processing times $p_j \in \{1, 2, 3\}$ on two machines is a $\frac{4}{3}$ -approximation algorithm for the makespan.*

Proof: Recall that \widetilde{M} denotes the minimum weight perfect matching found by the algorithm in the auxiliary graph \widetilde{G} . Let T^* denote a schedule with minimum makespan.

$$\begin{aligned}
\text{makespan}(T_{\widetilde{M}}) &= \frac{1}{2}(p(J) + |\text{holes}(T_{\widetilde{M}})|) \\
&= \frac{1}{2} \cdot (p(J) + w(\widetilde{M})) && \text{(by Lemma 4.3.2)} \\
&\leq \frac{1}{2} \cdot (p(J) + w(M_{T^*})) && \text{(since } \widetilde{M} \text{ is a min. weight matching)} \\
&\leq \frac{1}{2} \cdot \left[p(J) + \frac{1}{3} \cdot p(J) + \frac{4}{3} \cdot |\text{holes}(T^*)| \right] && \text{(by Lemma 4.3.3)} \\
&= \frac{4}{3} \cdot \frac{1}{2} [p(J) + |\text{holes}(T^*)|] \\
&= \frac{4}{3} \cdot \text{makespan}(T^*).
\end{aligned}$$

■

4.3.5 Tightness

The analysis of this algorithm is tight. Consider the following example in which the agreement graph is a complete bipartite graph $G = (J_1 \cup J_3, E)$, where J_1 consists of $3k$ unit jobs and J_3 has k triple jobs. An optimal schedule with no holes consists of segments in which a job in J_3 is scheduled concurrently with three jobs in J_1 . The makespan of the optimal schedule is therefore $3k$. The algorithm can only schedule two jobs in J_1 (and a hole) concurrently with each job in J_3 . The remaining k jobs in J_1 are scheduled as singletons. This results in a makespan of $4 \cdot k$, and the approximation ratio is $4/3$.

Chapter 5

A Reduction to k -Set-Cover

5.1 Unit Jobs

Consider the problem of scheduling unit jobs with conflicts on m machines, where m is constant. We present a reduction of this problem to finding a minimum set cover for the case in which all sets have cardinality at most m (in short, m -set-cover).

Let $G = (J, E)$ denote the conflict graph of a scheduling problem with unit jobs on m machines. We reduce G to a set system $\langle \mathcal{S}, U \rangle$ in which every set contains at most m elements as follows. The set of elements U is simply J . The collection of subsets \mathcal{S} is the set of all cliques of size at most m in the agreement graph \overline{G} . Note that if $A \in \mathcal{S}$, then every subset $A' \subset A$ is also in the collection \mathcal{S} . There is a one-to-one correspondence between covers $\mathcal{S}' \subseteq \mathcal{S}$ of U by disjoint sets and schedules T of jobs in J using m machines (here we ignore the order of rounds in a schedule and regard two schedules to be the same if one can be obtained from the other only by permuting the order of the rounds and the assignment of jobs to machines within each round). This correspondence maps a disjoint cover \mathcal{S}' to a schedule $T_{\mathcal{S}'}$ such that the makespan of $T_{\mathcal{S}'}$ equals the cardinality of \mathcal{S}' .

The k -set-cover problem has an approximation algorithm whose ratio equals $\mathcal{H}_k - \frac{1}{2}$ [DF97]. The reduction to k -set-cover implies that scheduling unit jobs on k machines is not harder to approximate than k -set-cover.

Claim 5.1.1 *The problem of scheduling unit jobs with conflicts on m machines (where m is constant) has an approximation algorithm whose approximation ratio $\mathcal{H}_m - \frac{1}{2}$. In particular, for $m = 3$, a $\frac{4}{3}$ -approximation exists.*

We remark that the tight example in [DF97] for $m = 3$ also holds for scheduling unit

jobs with conflicts. It remains open whether minimum makespan scheduling of unit jobs with conflicts on m machines is as hard to approximate as m -set-cover (SC_m).

5.2 Arbitrary Processing Times

The following lemma, for the case of jobs with varying sizes, is proved by scaling.

Lemma 5.2.1 *The problem of scheduling jobs with conflicts on m machines (where m is constant) has an approximation algorithm whose approximation ratio $2 \cdot \lceil \log_2 \frac{\max_j p_j}{\min_j p_j} \rceil \cdot (\mathcal{H}_m - \frac{1}{2})$.*

Proof: The algorithm, which we denote by Scaled- m -Set-Cover (SSC_m), works as follows. First it divides the jobs into bins J_i such that $j \in J_i$ if $2^{i-1} \leq p_j \leq 2^i$, for $i = \lceil \log_2(\min_j p_j) \rceil + 1, \dots, \lceil \log_2(\max_j p_j) \rceil$ (a job j with $p_j = 2^i$ that can be in J_i or in J_{i+1} , will arbitrarily be in one of them). Denote by \tilde{J}_i the set of jobs in J_i when the size of each is rounded down to 2^{i-1} . Namely, for each $j_\ell \in J_i$ there is a job \tilde{j}_ℓ of size 2^{i-1} in \tilde{J}_i , where the edges between the jobs are as in the original agreement graph. For each bin J_i , SSC_m runs m -set-cover on \tilde{J}_i . Let $\text{SC}_m(\tilde{J}_i)$ be the corresponding schedule. For each subset $\{\tilde{j}_1, \dots, \tilde{j}_t\}$ of $\text{SC}_m(\tilde{J}_i)$, SSC_m schedules $\{j_1, \dots, j_t\}$ concurrently (in time at most 2^i). For an illustration, see Figure 5.1.

Recall that $\text{OPT}(J_i)$ is an optimal makespan for J_i . We now bound the makespan of the schedule $\text{SSC}_m(J_i)$.

$$\begin{aligned}
|\text{SSC}_m(J_i)| &\leq 2 \cdot |\text{SC}_m(\tilde{J}_i)| && \text{(by rounding)} \\
&\leq 2 \cdot (\mathcal{H}_m - \frac{1}{2}) \cdot |\text{OPT}(\tilde{J}_i)| && \text{(approximation of } m\text{-set-cover)} \\
&\leq 2 \cdot (\mathcal{H}_m - \frac{1}{2}) \cdot |\text{OPT}(J_i)| && \text{(rounding down only reduces the makespan)}
\end{aligned}$$

Let $i_{\min} \triangleq \lceil \log_2(\min_j p_j) \rceil$ and $i_{\max} \triangleq \lceil \log_2(\max_j p_j) \rceil - 1$. The makespan of SSC_m is the accumulation of all the schedules $\text{SSC}_m(J_i)$ for $i = i_{\min}, \dots, i_{\max}$.

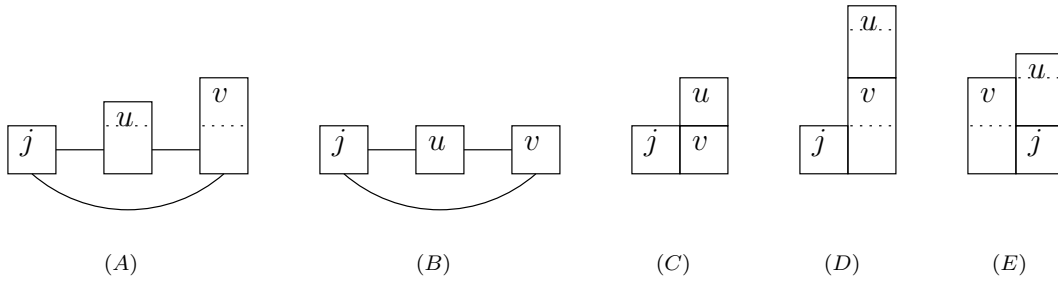


Figure 5.1: The figure shows an example of the reduction in the proof of Lemma 5.2.1. (A) An agreement graph \bar{G} with three jobs j, u, v where $p_j = 1 + \epsilon$ for a small $\epsilon > 0$, $p_u = 1.5$, $p_v = 2$. (B) The agreement graph over \tilde{J} (after rounding down the jobs). In this example all three jobs are in the same bin since their processing times is between 2^0 and 2^1 . (C) A schedule produced by the approximation algorithm of m -set-cover over \tilde{J}_i . (D) The schedule over J induced by the schedule over \tilde{J} . (E) An optimal schedule over J .

$$\begin{aligned}
 |\text{SSC}_m| &= \sum_{i=i_{\min}}^{i_{\max}} |\text{SSC}_m(J_i)| \\
 &\leq \sum_{i=i_{\min}}^{i_{\max}} 2 \cdot \left(\mathcal{H}_m - \frac{1}{2}\right) \cdot |\text{OPT}(J_i)| \\
 &\leq \sum_{i=i_{\min}}^{i_{\max}} 2 \cdot \left(\mathcal{H}_m - \frac{1}{2}\right) \cdot |\text{OPT}| \\
 &= 2 \cdot (i_{\max} - i_{\min} + 1) \cdot \left(\mathcal{H}_m - \frac{1}{2}\right) \cdot |\text{OPT}| \\
 &= 2 \cdot \left\lceil \log_2 \frac{\max_j p_j}{\min_j p_j} \right\rceil \cdot \left(\mathcal{H}_m - \frac{1}{2}\right) \cdot |\text{OPT}|
 \end{aligned}$$

■

Bibliography

- [Alo83] N. Alon. A note on the decomposition of graphs into isomorphic matchings. *Acta Mathematica Hungarica*, 42:221–223, 1983.
- [BC96] B. S. Baker and E. G. Coffman, Jr. Mutual exclusion scheduling. *Theoretical Computer Science*, 162(2):225–243, 1996.
- [BHK⁺00] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, R. Salman, and H. Shachnai. Sum multicoloring of graphs. *Journal of Algorithms*, 37:422–450, 2000.
- [BJ93] H. L. Bodlaender and K. Jansen. On the complexity of scheduling incompatible jobs with unit-times. In *MFCS '93: Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, pages 291–300, London, UK, 1993. Springer-Verlag.
- [BJ95] H. L. Bodlaender and K. Jansen. Restrictions of graph partition problems. Part I. *Theoretical Computer Science*, 148:93–109, 1995.
- [BJW94] H. L. Bodlaender, K. Jansen, and G. J. Woeginger. Scheduling with incompatible jobs. *Disc. Appl. Math.*, 55:219–232, 1994.
- [BKK⁺04] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. OPT versus LOAD in dynamic storage allocation. *SIAM Journal on Computing*, 33:632–646, 2004.
- [CGJL85] E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. Scheduling file transfers. *SIAM Journal on Computing*, 14(3):744–780, 1985.
- [DF97] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 256–264, 1997.

- [Dro96] M. Drozdowski. Scheduling multiprocessor tasks - an overview. *European Journal of Operational Research*, 94:167–191, 1996.
- [Edm65] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Standards Sect. B*, 69:125–130, 1965.
- [EHKR] G. Even, M. Halldórsson, L. Kaplan, and D. Ron. Scheduling with conflicts. full version available from <http://www.eng.tau.ac.il/~danar/papers.html>.
- [EL06] L. Epstein and A. Levin. On bin packing with conflicts. In *Proceedings of the 4th Workshop on Approximation and Online Algorithms (WAOA)*, pages 160–173, 2006.
- [FK98] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *JCSS*, 57:187–199, October 1998.
- [Gab90] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.
- [GG75] M. R. Garey and R. L. Graham. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4:187–200, 1975.
- [GJ75] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4:397–411, 1975.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman, 1979.
- [HHK93] P. Hansen, A. Hertz, and J. Kuplinsky. Bounded vertex colorings of graphs. *Discrete Math.*, 111(1-3):305–312, 1993.
- [HKP⁺03] M. M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle. Multicoloring trees. *Inf. Comput.*, 180(2):113–129, 2003.
- [IL03] S. Irani and V. Leung. Scheduling with conflicts on bipartite and interval graphs. *J. of Scheduling*, 6(3):287–307, 2003.

- [Jan99] K. Jansen. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3(4):363–377, 1999.
- [Jan03] K. Jansen. The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation*, 180(2):71–81, 2003.
- [JP02] K. Jansen and L. Porkolab. Polynomial time approximation schemes for general multiprocessor job shop scheduling. *Journal of Algorithms*, 45:167–191, 2002.
- [KGS95] D. Kaller, A. Gupta, and T. Shermer. The χ_t coloring problem. In *Symposium on Theoretical Aspects of Computer Science, STACS 95*, volume 900 of *Lecture Notes in Computer Sciences*. Springer, 1995.
- [Lon92] Z. Lonc. On complexity of some chain and antichain partition problems. In *WG '91: Proceedings of the 17th International Workshop*, pages 97–104, London, UK, 1992. Springer-Verlag.
- [Pet94] E. Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.
- [Zuc06] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 681–690, 2006.