Scheduling via SOS Hierarchies

Jurgen Aliaj

March 23 2018

1 SOS hierarchy review

1.1 Linear algebra preliminaries

We say that a matrix, $M \in \mathbb{R}^{n \times n}$, is *positive semidefinite* (PSD) if for every $x \in \mathbb{R}^n$, we have $x^T M x \geq 0$.

A $k \times k$ principle submatrix, N, of M is a submatrix obtained by choosing only the rows and columns of M corresponding to some subset of indices, $S \subset [n]$, of size k.

Fact 1. Every principle submatrix of a PSD matrix is PSD.

Proof. Let M be an $n \times n$ PSD matrix and let N be a principle submatrix of M indexed by some set S of size k. For any $x \in \mathbb{R}^k$, let $y \in \mathbb{R}^n$ be the vector such that $y_i = 0$ for every $i \notin S$ and the remaining entries in y match those of x. Then it's not hard to see that $x^T N x = y^T M y$, and the latter is non-negative by PSDness of M.

Fact 2. If M is a symmetric PSD matrix, every principal submatrix of M has non-negative determinant.

Proof. Let N be a principle submatrix of M, then by the previous fact we know N is PSD. Moreover, it's not hard to see that any principle submatrix of a symmetric matrix is also symmetric. Thus, we may use the Cholesky decomposition to see that $N = V^T V$ for some appropriate matrix V, and we have $|N| = |V|^2 \geq 0$.

With these definitions and facts, we are ready to define the SOS hierarchy.

1.2 Definition of SOS

Let $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ be some polytope where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$. It will be useful to think of K as being obtained from a linear relaxation of a binary optimization problem. Our ultimate (and unreachable) goal is to have $x \in conv(K \cap \{0,1\}^n)$, the integral hull. To do this we add additional variables and constraints to make the relaxation stronger. We add more and more variables and constraints at each level of the hierarchy to tighten the polytope and remove bad solutions, hopefully getting closer and closer to the integral hull the higher we go up in the hierarchy.

The t^{th} level of the SOS hierarchy, denoted as $\text{SOS}_t(K)$, is defined as the set of vectors $y \in \mathbb{R}^{2^{[n]}}$ satisfying the following properties.

- $M_t(y) := (y_{I\cup J})_{|I|,|J|\leq t} \succeq 0$
- $M_t^\ell := \left(\sum_{i=1}^n A_{\ell i} y_{I\cup J\cup\{i\}} - b_\ell y_{I\cup J}\right)_{|I|,|J|\leq t} \succeq 0 \qquad \forall \ell \in [m]$
- $y_\varnothing = 1$

Here the vectors y are indexed by subsets of $[n]$ corresponding to subsets of the original variables $x_1, \ldots, x_n$. We would like to think of the vector y as a probability distribution. More precisely, for any $X \in K \cap \{0,1\}^n$ drawn from this distribution we would like the variable y_I to represent the probability $Pr[\bigwedge_{i\in I}(X_i = 1)]$. In particular, we would like $y_i = x_i$ and $y_\varnothing = 1$. The matrix $M_t(y)$ is known as the *moment matrix* of y and here we impose that it is PSD. The matrix $M_t^\ell(y)$ is the *moment matrix* of *slacks* corresponding to constraint ℓ and we similarly impose PSDness for each ℓ. Intuitively, the constraint $M_t(y) \succeq 0$ forces the variables to be consistent (for example, it guarantees $y_{\{1,2\}} \in [\max(y_{\{1\}} + y_{\{2\}} - 1, 0), \min(y_{\{1\}}, y_{\{2\}})]$). Whereas PSDness of the moment matrix of slacks guarantees that the solution y satisfies the original constraints. Also observe that only variables indexed by subsets of size at most $2t+1$ are mentioned by the t^{th} level of SOS, so one can optimize over $\text{SOS}_t(K)$ in time $m^{O(1)}n^{O(t)}$ up to numerical errors that can be neglected for our purposes.

Finally, we define $\text{SOS}_t^{proj}(K) := \{(y_{\{1\}}, \ldots, y_{\{n\}}) \mid y \in \text{SOS}_t(K)\}$ as the projection onto the original variables.

1.3 Some useful properties

Lemma 1. Let $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ and $y \in \text{SOS}_t(K)$ for $t \geq 0$. Then the following holds:

a) $K \cap \{0,1\}^n \subseteq \text{SOS}_t^{proj}(K)$

b) $0 \leq y_I \leq 1$ for all $|I| \leq t$

c) $y_J \leq y_I$ for all $I \subseteq J$ with $|I|, |J| \leq t$

d) $|y_{I\cup J}| \leq \sqrt{y_I y_J}$ for $|I|, |J| \leq t$

e) $\text{SOS}_t^{proj}(K) \subseteq K$

f) $\text{SOS}_0(K) \supseteq \text{SOS}_1(K) \supseteq \cdots \supseteq \text{SOS}_n(K)$

Proof.

a) Let $x \in K \cap \{0,1\}^n$ and define $y_I := \prod_{i\in I} x_i$. It suffices to show $y \in \text{SOS}_t(K)$ for every $t \geq 0$. First note that the moment matrix $M_t(y)$ is a submatrix of the matrix yy^T. This is because the entry in yy^T indexed by I, J is $y_I y_J = \left(\prod_{i\in I} x_i\right) \cdot \left(\prod_{i\in J} x_i\right) = \left(\prod_{i\in I\cup J} x_i\right)$. This last equality follows from the fact that $x_i^2 = x_i$ for $x_i \in \{0,1\}$. Moreover, this term is precisely $y_{I\cup J}$ which is the entry

in $M_t(y)$ indexed by I, J. Now notice that yy^T is a PSD matrix since for any $a \in \mathbb{R}^{2^{[n]}}$ we have $a^T yy^T a = (a^T y)^2 \geq 0$. So by fact 1, we have that $M_t(y)$ is PSD. We can do something similar for $M_t^\ell(y)$. Consider the ℓ^{th} constraint $A_\ell \cdot x \geq b_\ell$, then I claim $M_t^\ell(y)$ is a submatrix of $(A_\ell \cdot x - b_\ell)yy^T$. This matrix is also PSD since yy^T is PSD and multiplying a matrix by a non-negative constant preserves PSDness (the constant is non-negative using the fact that x is feasible). The entry of $M_t^\ell(y)$ indexed by I, J is $\sum_{i=1}^n A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J} = (A_\ell \cdot x - b_\ell) y_I y_J$. Here I am simply factoring $y_{I \cup J} = y_I y_J$ again using the fact that the values of x are integral, and using the fact $y_{\{i\}} = x_i$.

b) Consider the principal submatrix of $M_t(y)$ indexed by $\{\varnothing, I\}$. This matrix is:

$$\begin{bmatrix} y_\varnothing & y_I \\ y_I & y_I \end{bmatrix}$$

The determinant of this submatrix is $y_I(1 - y_I)$, and since $M_t(y)$ is symmetric we can use fact 2 to see that this value is non-negative. So $0 \leq y_I \leq 1$.

c) Consider the principal submatrix of $M_t(y)$ indexed by $\{I, J\}$. This matrix is:

$$\begin{bmatrix} y_I & y_{I \cup J} \\ y_{I \cup J} & y_J \end{bmatrix}$$

Here we use the fact that $I \cup J = J$ hence the determinant of this submatrix is $y_J(y_I - y_J) \geq 0$. Now by part b), y_J is non-negative so the result follows.

d) Once again consider the principal submatrix of $M_t(y)$ indexed by $\{I, J\}$. This matrix is:

$$\begin{bmatrix} y_I & y_{I \cup J} \\ y_{I \cup J} & y_J \end{bmatrix}$$

The determinant of this submatrix is $y_I y_J - y_{I \cup J}^2 \geq 0$. The result follows.

e) For each moment matrix of slacks, the entry indexed by $\varnothing, \varnothing$ forces the vector $(y_1, \ldots, y_n)$ to satisfy the appropriate constraints.

f) This follows from the fact that $M_t(y)$ is a principal submatrix of $M_{t+1}(y)$. The same holds true for the slack matrices.

Finally we present a useful lemma which allows us to write a feasible solution in $\text{SOS}_t(K)$, y, as a convex combination of vectors.

Lemma 2. For $t \geq 1$, let $y \in \text{SOS}_t(K)$ and $i \in [n]$ be a variable with $0 < y_i < 1$. If we define

$$z_I^{(1)} := \frac{y_{I \cup \{i\}}}{y_i} \quad \text{and} \quad z_I^{(0)} := \frac{y_I - y_{I \cup \{i\}}}{1 - y_i} \tag{1}$$

then we have $y = y_i \cdot z^{(1)} + (1 - y_i) \cdot z^{(0)}$ with $z^{(0)}, z^{(1)} \in \text{SOS}_{t-1}(K)$ and $z_i^{(0)} = 0, z_i^{(1)} = 1$.

Verifying that y can be written in this form is obvious. The second part of the statement (verifying that $z^{(0)}, z^{(1)} \in \mathrm{SOS}_t(K)$) is less obvious, but one can see the proof in the lecture notes by Thomas Rothvoss. Moreover, notice that the vector $z^{(1)}$ preserves zeroes (that is, if $y_j = 0$ then $z_j^{(1)} = 0$). This can be seen by part d) of lemma 1. In fact, both vectors preserve all integral values.

2 Scheduling on two identical machines with precedence constraints

In the scheduling problem, we are given a set, J, of n jobs with unit processing time as well as $m = 2$ identical machines. We are also given a set S of precedence constraints of the form $i \prec j$, which indicates that job i must finish before job j can be started. The goal is to schedule the jobs on the machines so that the makespan (the time of the last scheduled job) is minimized.

2.1 A simple algorithm based on SOS hierarchies

In this section we give a brief application of SOS hierarchies by presenting an algorithm that only uses solutions from the first level of the hierarchy.

We can solve the scheduling problem with 2 machines with a straightforward integer program. We consider T time slots of unit length and we would like to see if we can find a feasible schedule with makespan T. The IP is given below:

$$\begin{aligned}
&\sum_{t=1}^{n} x_{jt} = 1 && \forall j \in J \\
&\sum_{j \in J} x_{jt} \leq 2 && \forall t \in [T] \\
&\sum_{t' \leq t} x_{it'} \geq \sum_{t' \leq t+1} x_{jt'} && \forall i \prec j, \forall t \in [T] \\
&x_{jt} \in \{0,1\} && \forall j \in J, \forall t \in [T]
\end{aligned} \tag{2}$$

The variable x_{jt} indicates whether or not job j will be scheduled in the interval $[t-1, t]$. The first constraint takes care that each job is scheduled precisely once. The second constraint takes care of the fact that we cannot schedule on more than 2 machines, and the third constraint takes care of the precedence constraints. Now we relax the integrality constraints by letting $x_{jt} \geq 0$ and we let $K(T)$ represent the feasible solution of the LP relaxation. We give an instance of the problem where the LP relaxation fails on the next page.

On the other hand, we claim that we can solve the problem optimally using only the first level of the SOS hierarchy. More specifically, if there exists $y \in \mathrm{SOS}_1(K(T))$, then there is also a feasible schedule $\sigma : J \to [T]$. Conversely, if a feasible schedule exists then there is a feasible integral solution to (2), and by part a) of the first lemma we will be able to find such a y. To find the optimal value of T, we can then do a binary search for T in the range $\{\lceil n/2 \rceil, \ldots, n\}$.

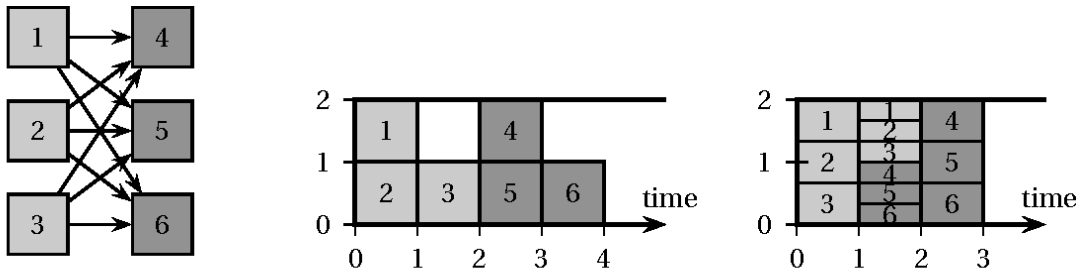

Figure 1: The instance is represented in the first image, where an arrow (i, j) indicates $i \prec j$. The second image shows an optimal schedule with makespan 4. The third image shows a feasible fractional solution with makespan 3, so the integrality gap is at least 4/3.

Consider the following procedure. Given $y \in \mathrm{SOS}_1(K(T))$, define the *fractional completion time* $C_j^* := \max\{t \mid y_{j,t} > 0\}$. Sort the jobs so that $C_1^* \leq C_2^* \leq \cdots \leq C_n^*$. Now go through the time slots from 1 to T and for each one, choose the two lowest index jobs among the set of jobs that are unprocessed and have all dependent jobs processed. It may be impossible to choose two jobs for a given time slot (for example if every job depends on, say, the first job). In this case we simply schedule one job in the time slot. We prove a final lemma which immediately implies correctness of our procedure.

Lemma 3. For any job $j \in J$ we have $\sigma_j \leq C_j^*$.

Proof. Let j_1 be the lowest index job that does not satisfy the claim. Let $j_0 \in \{1, \ldots, j_1 - 1\}$ be the last job scheduled without any other job in $\{1, \ldots, j_1\}$ in parallel. If such a job does not exist, we can introduce a dummy job j^* which every other job depends on. This job will then be scheduled alone in the first time slot.

Let $J_0 := \{j \in J \mid j \leq j_1 \text{ and } \sigma_j > \sigma_{j_0}\}$ and notice that every job in J_0 depends on j_0. This is because if $j \in J_0$ does not depend on j_0 then we could have scheduled it concurrently with j_0 (j_0 is scheduled with no other job or with a job with index higher than j_1). By the choice of j_0, the interval $[\sigma_{j_0}, \sigma_{j_1} - 1]$ is fully busy with jobs from J_0. Suppose this interval has length k, then $|J_0| \geq 2k$. Moreover, $j_1 \in J_0$ hence $|J_0| > 2k$ is a strict inequality.

Now we find a solution $\widetilde{y} \in \mathrm{SOS}_0(K)$ such that $\widetilde{y}_{j_0 C_{j_0}^*} = 1$. If $y_{j_0 C_{j_0}^*} = 1$ already we can simply let $\widetilde{y} = y$. Otherwise we have $y_{j_0 C_{j_0}^*} \in (0, 1)$ and we can apply lemma 2 with the vector y and $i = j_0, C_{j_0}^*$. In particular, we let $\widetilde{y} = z^{(1)}$ defined in (1). Now since this variable is 1, it forces the fractional schedule $\widetilde{y}$ to schedule all dependent jobs in J_0 later than $C_{j_0}^*$ (by the 3^{rd} set of constraints in (2)), and this value is at least σ_{j_0} (by minimality of j_1). Moreover, $C_j^* \leq \sigma_{j_1} - 1$ for all $j \in J_0$ by assumption. So we must have $y_{j,t} = 0$ for all $j \in J_0$ and $t > \sigma_{j_1} - 1 \geq C_j^*$ (by definition of C_j^*). So therefore the same must hold true for the fractional schedule $\widetilde{y}$, since it preserves zeroes.

So in summary, we have $\widetilde{y}_{j,t} = 0$ for all $j \in J_0$ and $t < \sigma_{j_0}$ or $t > \sigma_{j_1} - 1$. Thus the fractional schedule must schedule the entire set of jobs J_0 in the interval $[\sigma_{j_0}, \sigma_{j_1} - 1]$. But this is a contradiction since this interval has only $2k$ slots and J_0 has strictly more than $2k$ jobs. □

2.2 A polynomial time algorithm from 50 years ago

I should now point out that this problem was already solved optimally in polynomial time nearly 50 years ago by Coffman & Graham (1972). We suppose an instance of the problem is given as a directed graph (for example, see the instance in figure 1). Given an ordered list L of vertices (representing jobs), we obtain a schedule as follows. For each time step t, both machines try to schedule a job with the lowest index in L such that all predecessors of the job have been processed. If both machines try to schedule the same job we make the convention that the first machine schedules the job while the second machine will look for the job with the second lowest index that satisfies this property (it may be the case that the second machine may not be able to find such a job). Our algorithm in section 2.1 is essentially this procedure with a suitably chosen L. Each ordered list L yields a schedule with makespan $\omega(L)$, and the goal is to find an order L^* such that $\omega(L^*) \leq \omega(L)$ for every L. The interested reader is invited to see Coffman & Graham (1972, p. 203) for the details of how L is chosen.

2.3 Recent scheduling results via LP hierarchies

Fortunately, there are some state of the art results in scheduling using the method of LP hierarchies. These results use similar lift and project methods, and they lift the same LP given in (2).

For example, for a fixed number of machines m and fixed ϵ, Levey & Rothvoss (2016) give a $(1+\epsilon)$-approximation for the scheduling problem using r rounds of Sherali-Adams, thus yielding a running time of $n^{O(r)}$. Here $r = (\log n)^{\Theta(\log \log n)}$.

More recently, the result above was improved to $r = (\log n)^{O(1)}$ by Garg (2017), yielding a quasi-PTAS.

2.4 Open problems

For scheduling with $m = 3$ machines and precedence constraints, it is unknown if the problem is NP-hard. Moreover, a PTAS is not known, but it is suspected that one exists based on LP or SDP hierarchies.

References

[1] Coffman, E. G., & Graham, R. L. (1972). Optimal scheduling for two-processor systems. *Acta informatica*, 1(3), 200-213.

[2] Garg, S. (2017). Quasi-PTAS for Scheduling with Precedences using LP Hierarchies. *arXiv preprint arXiv:1708.04369*.

[3] Levey, E., & Rothvoss, T. (2016, June). A $(1+\epsilon)$-approximation for makespan scheduling with precedence constraints using LP hierarchies. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing* (pp. 168-177). ACM.

[4] Rothvoss, T. (2013). The Lasserre hierarchy in approximation algorithms. *Lecture Notes for the MAPSP*, 1-25.