SEAMS school on optimization and algorithms in dynamic environments

by Christoph Dürr (scribe)

February 2019, Hanoi, Vietnam

1 Standard tools: doubling and linear programming

We use the same framework of competitive analysis to analyze both online algorithms and related models of optimization with hidden information, such as the cow path problem or online bidding.

In the online algorithmic framework, the input is provided to the algorithm in form of a sequence of requests. Every request has to be answered, without knowing the future requests. For a particular optimization problem which we have in mind, say a cost minimization problem, we use I to denote an instance, $\mathrm{ALG}(I)$ the objective value of the solution produced by the algorithm, and $\mathrm{OPT}(I)$ the value of the optimal offline solution. By the latter we mean a solution which we could have been computed if we knew all the instance in advance. The competitive ratio of the algorithm, denoted C.R.(ALG) is the suppremum of $\mathrm{ALG}(I)/\mathrm{OPT}(I)$ taken over all instances. And the competitive ratio of the problem is the infinum of C.R.(ALG) taken over all algorithms. Here we emphase that we do not restrict to polynomial time algorithms for simplicity. An algorithm can be any function, that maps a prefix of an instance — which represents the portion of the input the algorithm gets to see — to a decision. But don't worry, most online algorithms are quite simple and have small running times.

1.1 The cow path problem

[more material can be found in the lecture notes by Shahin Kamali, notes 1 and handouts 2 at http://www.cs.umanitoba.ca/~kamalis/fall18/]

This problem has been introducted by Bellman and Beck (1963) in a Bayesian context, and has been rediscovered later. It is also called *linear search*.

We are given a straight fence represented as a line. There is a cow on one side of the fence at position 0, and juice grass on the other side of the fence. There is a single opening at some position u (which stands for $u$nknown). For technical reasons we assume $|u| \geqslant 1$, otherwise we have to deal with infinite small movements of the cow. The cow knows only the setting, but not u nor its sign. A strategy for the cow is defined by a sequence of numbers $d_0, d_1, \ldots$ with $d_0 \geqslant 1$ and $d_{k-2} \leqslant d_k$. The idea is that the algorithm walks distance d_0 say to the left, then back, then distance d_1 right from the origin, and then back, and so on, until it reaches position u. To simplify the notation we assume that u is positive, and that the algorithms starts with the worst possible direction. Suppose that he reaches the opening during its k-th walk, which means that $d_{k-2} < u \leqslant d_k$. His competitive ratio is

$$\frac{2d_0 + 2d_1 + \ldots + 2d_{k-1} + u}{u}.$$

So the ratio is maximized when u is as close as possible to d_{k-2}, say $u = d_{k-2} - \varepsilon$ for an arbitrary small $\varepsilon > 0$. One possible optimal strategy is to choose $d_i = 2^i$. This strategy is called *doubling*. After some computation we find out that with this choice the ratio is arbitrary close to 9. There is a matching lower bound for any deterministic online algorithm.

Randomization helps

A randomized strategy is a distribution over deterministic strategies, and by its cost we mean the expected cost. First step is to start uniformly with left or right, and then alternate as described above using the doubling strategy. The ratio of this strategy drops to 7. One additional improvement can be obtained increasing at each step not with factor 2, but by a bigger factor $r \approx 3.591$, which is the solution to $r \ln r = r + 1$. In addition the first step is randomized, d_0 is choosen to be r^x for some uniform random value $x \in [0, 1]$. The resulting ratio is 1+r=4.591 and there is a matching lower bound for all randomized strategies.

1.2 Exercice: online bidding

There is an unknown value u, with the property $u \geqslant 1$. The algorithm needs to guess an upper bound on u, and pays for each guess. This means that the algorithm needs to generate a sequence of increasing numbers $x_0, x_1, x_2, ...$ His cost is $x_0 + ... + x_k$ for the first index k such that $x_k \geqslant u$. Hence the ratio is

$$\frac{x_0 + ... + x_k}{u},$$

which is maximized for $u = x_{k-1} + \varepsilon$ for an arbitrary small $\varepsilon > 0$. The algorithm can minimize this ratio by choosing $x_i = 2^i$, leading to a factor of 4. This is optimal for deterministic algorithms and the randomized competitive ratio is e.

For more information, read M. Chrobak, C. Kenyon, J. Noga, and N. E. Young. *Incremental medians via online bidding*. Algorithmica, 50(4):455–478, 2008.

1.3 A super quick introduction in linear programming and duality

[from wikipedia] The founders of this subject are Leonid Kantorovich, a Russian mathematician who developed linear programming problems in 1939, Dantzig, who published the simplex method in 1947, and John von Neumann, who developed the theory of the duality in the same year.

Formally we have to find values for n variables $x_1, ..., x_n$, minimizing $\sum_{i=1}^{n} c_i x_i$ such that for all $j = 1, ..., m$ have $\sum_{i=1}^{n} a_{ji} x_i \geq b_j$, and in addition $x_1, ..., x_n \geqslant 0$. This is called the standard form of an LP. Variables without restrictions on their sign can be modelled by the difference of two non-negative variables. Constraints with $\leqslant$ can be modelled by multiplying by -1 a constraint with $\geqslant$. Constraints with $=$ can be simulated with two constraints $\leqslant$ and $\geqslant$. Maximization problems can be modelled by multiplying the objective value by -1.

A linear program could infeasible, unbounded (meaning you can make the objective as small as desired) and otherwise it is a finite optimum. It can be solved in time polynomial in n and m. When adding integrality constraints on some variables, which then typically encode some 0/1 decision for a combinatorial optimization problem, the model becomes a so called *integer linear program*. Solving them however is in general a hard problem (NP-hard for those who know the term).

[more details can be found in the book *Approximation algorithms* by Vijay Vazirani]

1.3.1 Duality explained on an example

Consider the following LP.

$$\begin{aligned} & \min & & 7x_1 + x_2 + 5x_3 \\ s.t. & & & x_1 - x_2 + 3x_3 \geqslant 10 \quad (1) \\ & & & 5x_1 + 2x_2 - x_3 \geqslant 6 \quad (2) \\ & & & x_1, x_2, x_3 \geqslant 0. \end{aligned}$$

How can be obtain a lower bound on the optimum? Observe the first constraint. The coefficient of every variable in the left hand side is a lower bound on the corresponding coefficient in the objective value. Hence since the variables are non-negative we have

$$7x_1 + x_2 + 5x_3 \geqslant x_1 - x_2 + 3x_3 \geqslant 10.$$

But summing up the two equations we obtain

$$6x_1 + x_2 + 2x_3 \geqslant 16,$$

which satisfies the same observation, hence we have the stronger lower bound 16. The strongest lower bound can be found by taking a linear combination of the two constraints. Call y_1, y_2 the coefficients. Then the strongest lower bound is the solution to this LP.

$$\begin{aligned} & \max & 10y_1 + 6y_2 \\ s.t. & & y_1 + 5y_2 \leqslant 7 \\ & & -y_1 + 2y_2 \leqslant 1 \\ & & 3y_1 - y_2 \leqslant 5 \\ & & y_1, y_2 \geqslant 0. \end{aligned}$$

See, how the right hand sides of the constraints and the coefficients in the objective value got exchanged, and how *min* became *max*, and also how $\leqslant$ became $\geqslant$. It is like looking through a mirror. We call this LP the dual, and the first one the primal. We have

Theorem 1. (Duality theorem)

The primal is infeasible iff the dual is unbounded.

The primal is unbounded iff the dual is infeasible.

Otherwise the optimum to the primal equals the optimum to the dual.

By construction we have that the optimum to the primal is at most the optimum to the dual. This is known as the *weak duality*. The fact that this holds with equality is the *strong duality* and its proof would need to explain how the Simplex algorithm works.

If we write an LP in a more compact form using vectors b, c and a matrix A, we have that the dual to $\min\{c^T x \colon Ax \geqslant b, x \geqslant 0\}$ is $\max\{b^T y \colon A^T y \leqslant c, y \geqslant 0\}$, and vice-versa.

1.3.2 Complementary slackness

Let x be a solution to the primal of value P.

Let y be a solution to the dual of value D.

Let $s_j := \sum_i a_{ji} x_i - b_j$ the slack of the j-th primal constraint.

Let $t_i := c_i - \sum_j a_{ji} y_j$ the slack of the i-th dual constraint. Then we have

$$P - D = c^T x - b^T y = s^T y + t^T x.$$

You obtain the second inequality by plugging in the definitions of s and t. This means

Theorem 2. *The solutions x and y are optimal iff $s_j y_j = 0$ for all j and $t_i x_i = 0$ for all i. In other words iff the following so called complementary slackness conditions are satisfied:*

For every j either the j-th primal constraint is saturated (means satisfied with equality) or the j-th dual variable is 0.

For every i either the i-th dual constraint is saturated or the i-th primal variable is 0.

Here by *or* we don't mean the exclusive or. Both conditions could hold.

1.3.3 Primal dual algorithm

This algorithm will not only solve an LP but also provide a proof of optimality. Start with $x = 0$, $y = 0$. Invariant to the following loop is that y is feasible and that the complementary slackness condition holds. Now while x is not feasible, increase non frozen variables y, until some dual constraint becomes tight. Then we have the possibility to increase the corresponding primal variables in the hope to satisfy some primal constraints. Freeze all dual variables which are involved in tight constraints.

When the loop ends, we have a solution pair x, y, where x is the solution to the primal LP, while y serves as an optimality proof. How we change x and y is problem specific. As an example we consider the shortest path problem. We are given a directed graph $G(V, A)$ with a source $s \in V$, and arc lengths $\ell\colon A \to \mathbb{R}^+$. To make some physical analogy work we assume that for every arc (u, v) there is a corresponding arc (v, u) in the graph of same length. Now imagine a set of balls weighting 1 kg each, and representing the vertices. The balls u, v such that there is an arc (u, v) are connected by a rope of length ℓ(u,v). Now hold the construction at the ball s and let the gravity pull on the balls. The height of every ball v, compared to the height of s is the distance in the graph. The height represent dual variables in the following model. In addition, if on a string connecting u, v a force of a kg is pulling, this means that there are a shortest paths from the source s that traverse the corresponding arc (u, v). This force is represented by the primal variables.

$$\begin{aligned}
\text{primal}\quad \min\quad & \sum_{uv\in A} \ell(u,v)x_{u,v} \\
s.t.\quad & \forall v \in V\colon \sum_{uv\in A} x_{u,v} - \sum_{vu\in A} x_{v,u} = \begin{cases} 1 & \text{if } v \neq s \\ 1-|V| & \text{if } v = s. \end{cases} \\
& \forall uv \in A\colon x_{u,v} \geqslant 0
\end{aligned}$$

$$\begin{aligned}
\text{dual}\quad \max\quad & \sum_{v\in V\setminus\{s\}} (y_v - y_s) \\
s.t.\quad & \forall uv \in A\colon y_v - y_u \leqslant \ell(u,v). \\
& \forall v \in V\colon y_v \geqslant 0.
\end{aligned}$$

Now the primal dual algorithm can be viewed as simulating the dynamics of the balls starting all from the same position as s and falling down. Whenever some ball v is stopped in its fall, there is a tight path from s to v, and the last rope on that path become just now tight. The corresponding primal variable will be increased by 1 along the path. The dual variables y_v will be frozen.

1.3.4 Approximate version of the complementary slackness condition

Let α, β be some constants, and x, y respective a primal and dual solution.

If we have

- For every j either the j-th primal constraint is saturated up to factor α or the j-th dual variable is 0.
- For every i either the i-th dual constraint is saturated up to factor β or the i-th primal variable is 0.
- Formally, for every j we have $y_j = 0$ or $b_j \leqslant \sum_i a_{ji}x_i \leqslant \beta b_j$ and for every i we have $x_i = 0$ or $c_i/\alpha \leqslant \sum_j a_{ji}y_j \leqslant c_i$,

then x is a a solution that is at most $\alpha \cdot \beta$ times the optimal solution.

This theorem is useful for analyzing online and approximation algorithms.

1.4 Ski rental

A first trivial problem to illustrate the primal dual approach. You are going to ski. But you don't know in advance for how long, since you will break your leg at day $\ell+1$, but don't know ℓ. Every day you could rent the skis for 1 Euro or just buy them once for all for B Euros. The optimal deterministic strategy is to rent for the first $B-1$ days then to buy them on day B. If $\ell < B$ the algorithm is optimal, otherwise it has a ratio close to 2, $2-1/B$ to be precise.

$$\begin{aligned} \text{primal} \quad \min \quad & B \cdot x + \sum_{i=1}^{\ell} z_i \\ s.t. \quad & \forall i = 1 \ldots \ell : x + z_i \geqslant 1 \\ & x, z_i \geqslant 0 \end{aligned}$$

$$\begin{aligned} \text{dual} \quad \max \quad & \sum_{i=1}^{\ell} y_i \\ s.t. \quad & \sum_{i=1}^{\ell} y_i \leqslant B \\ & \forall i : y_i \leqslant 1 \\ & \forall i : y_i \geqslant 0. \end{aligned}$$

Again, start with all variables at zero. Every time i, a new primal and corresponding dual variable arrives. We raise y_i until some dual constraint becomes tight. Then we increase the corresponding primal variable. This means that the first $B-1$ time steps, we will only increase z_i. Then at day B the (x) dual constraint becomes tight and we raise x. For these LPs raising, means setting to 1.

We can check that

- If $y_i > 0$ then $1 \leqslant x + z_i \leqslant 2$. First holds by algorithm, and second by the fact that we don't increase variables above 1.
- If $x > 0$, then $\sum y_i = B$.
- If $z_i \geqslant 0$, then $y_i = 1$.

Hence the approximate complementary slackness conditions are satisfied with $\alpha = 1$ and $\beta = 2$, meaning the solution is a 2 approximation.

1.5 Set covering

You have sets with costs. But the elements arrive online. Then you learn to which sets the new element belongs. It has to be covered by at least one selected set. You pay for selecting a set. You can never unselect a set.

For an illustrative example, think that a set is a possible position for a school in a growing village. Children cannot commute too far, so there is a set of house locations associated to a possible school position. You observe houses being build, and need to open schools to cover them. Of course you will not destroy schools, only open new ones.

Let d be the maximum cardinality of the sets at the end. There is a lower bound of d on the competitive ratio of any deterministic algorithm. Just release d elements, each belongs to all non-selected sets. Optimum is 1, algorithm chooses d no matter how.

Randomization helps. One way is to consider first the fractional deterministic variant. Sets can be selected in fractions. But the sum of the fractions of the sets containing a given element should be at least 1. Later we can use some technique to turn a fractional solution into a randomized 0/1 solution. But it is not an easy technique, in my point of view.

Now for the fractional variant there is a lower bound of $O(\log\ d)$, and a primal dual online algorithm reaching this bound. For more details, read lecture note 3 by Nikhil Bansal at https://www.win.tue.nl/~nikhil/AU16/). And for complete details read the book *The Design of Competitive Online Algorithms via a Primal–Dual Approach* by Niv Buchbinder and Joseph (Seffi) Naor, Foundations and Trends in Theoretical Computer Science, Vol. 3, Nos. 2–3 (2007) 93–263.