# Supporting hyperplanes



Figure 1: Geometric interpretation of the dual with optimal value  $g(\lambda) = \beta_{\lambda}$ . In the situation on the left strong duality holds, and  $\beta_{\lambda} = \phi(b)$ . In the situation on the right, strong duality does not hold, and  $\beta_{\lambda} < \phi(b)$ .

# General approach when using Lagrangian methods

More generally, to

minimize 
$$f(x)$$
 subject to  $h(x) \le b, x \in X$ , (1.3)

we proceed as follows:

1. Introduce a vector z of slack variables to obtain the equivalent problem

minimize f(x) subject to  $h(x) + z = b, x \in X, z \ge 0$ .

- 2. Compute the Lagrangian  $L(x, z, \lambda) = f(x) \lambda^T (h(x) + z b)$ .
- 3. Define the set

$$Y = \{\lambda \in \mathbb{R}^m : \inf_{x \in X, z \ge 0} L(x, z, \lambda) > -\infty\}.$$

4. For each  $\lambda \in Y$ , minimize  $L(x, z, \lambda)$  subject only to the regional constraints, i.e. find  $x^*(\lambda), z^*(\lambda)$  satisfying

$$L(x^*(\lambda), z^*(\lambda), \lambda) = \inf_{x \in X, z \ge 0} L(x, z, \lambda).$$
(1.4)

5. Find  $\lambda^* \in Y$  so that  $(x^*(\lambda^*), z^*(\lambda^*))$  is feasible, i.e. so  $x^*(\lambda^*) \in X$ ,  $z^*(\lambda^*) \ge 0$ , and  $h(x^*(\lambda^*)) + z^*(\lambda^*) = b$ . By Theorem 1.2,  $x^*(\lambda^*)$  is optimal for (1.3).

### Lecture 1 homework

#### Homework

- 1. Consider  $P_b$  when f(x) = -x, and in two cases
  - (a)  $h(x) = \sqrt{x}, b = 2;$
  - (b)  $h(x) = x^2, b = 16.$

What happens when you try to solve these problems using Lagrangian methods? In each case, find  $\phi(b)$  and explain whether strong duality holds.

2. Given  $a_1, \ldots, a_n > 0$ ,

minimize 
$$-\sum_{i=1}^{n} \log(a_i + x_i)$$
  
subject to  $x_1, \dots, x_n \ge 0$  and  $\sum_i x_i = b$ .

The optimal x corresponds to one that can be found by a so-called 'water filling algorithm'. Imagine placing bars of heights  $a_i$  side by side in the fashion of a histogram and then flooding above these bars so as to cover area of b. Draw a picture to illustrate this idea.

### Shadow prices

**Theorem 2.5.** Suppose that f and h are continuously differentiable on  $\mathbb{R}^n$ , and that there exist unique functions  $x^* : \mathbb{R}^m \to \mathbb{R}^n$  and  $\lambda^* : \mathbb{R}^m \to \mathbb{R}^m$  such that for each  $b \in \mathbb{R}^m$ ,  $h(x^*(b)) = b$  and  $f(x^*(b)) = \phi(b) = \inf\{f(x) - \lambda^*(b)^T(h(x) - b) : x \in \mathbb{R}^n\}$ . If  $x^*$  and  $\lambda^*$  are continuously differentiable, then

$$\frac{\partial \phi}{\partial b_i}(b) = \lambda_i^*(b)$$

*Proof.* We have that

$$\phi(b) = f(x^*(b)) - \lambda^*(b)^T (h(x^*(b)) - b)$$
  
=  $f(x^*(b)) - \lambda^*(b)^T h(x^*(b)) + \lambda^*(b)^T b.$ 

Taking partial derivatives

$$\frac{\partial \phi(b)}{\partial b_i} = \sum_{j=1}^n \left( \frac{\partial f}{\partial x_j} (x^*(b)) - \lambda^*(b)^T \frac{\partial h}{\partial x_j} (x^*(b)) \right) \frac{\partial x_j^*}{\partial b_i} (b) - \frac{\partial \lambda^*(b)^T}{\partial b_i} (h(x^*(b)) - b) + \lambda^*(b)^T \frac{\partial b}{\partial b_i}.$$

The first term on the right-hand side is zero, because  $L(x, \lambda^*(b))$  is stationary with respect to  $x_j$  at  $x^*(b)$ . The second term is zero as well, because  $x^*(b)$  is feasible and thus  $(h(x^*(b)) - b) = 0$ . The claim follows.

## A linear programming problem

$$\min\left\{c^T x : Ax \ge b, x \ge 0\right\}$$





Figure 2: Geometric interpretation of the linear program of Example 2.3

# Lecture 3

The simplex algorithm is due to George Dantzig. In his paper on the origins of the simplex algorithm he writes:

"In the summer of 1947, when I first began to work on the simplex method for solving linear programs, the first idea that occurred to me is one that would occur to any trained mathematician, namely the idea of step by step descent (with respect to the objective function) along edges of the convex polyhedral set from one vertex to an adjacent one. I rejected this algorithm outright on intuitive grounds - it had to be inefficient because it proposed to solve the problem by wandering along some path of outside edges until the optimal vertex was reached. I therefore began to look for other methods which gave more promise of being efficient, such as those that went directly through the interior."

This is interesting because Dantzig says that the simplex method is something that any trained mathematician would think of - but that that on first sight it appears to be inefficient.

In practice these days computers solve problems with 100,000s constraints and variables. There is a nice on line <u>simplex method tool</u> which you might like to use to check answers that you obtain by hand when doing questions on Examples sheet 1. There is also a very nice program here, which runs under Windows.

http://trin-hosts.trin.cam.ac.uk/fellows/dpk10/IB/simple2x.html

It was written by Doug Kennedy at Trinity. It takes the hard labour out of performing pivot operations and it may be configured to prompt the choice of pivot elements, or to solve problems automatically.

# The simplex tableau

	m		1
	B	N	
$m \left\{ \right.$	$A_B^{-1}A_B = I$	$A_B^{-1}A_N$	$A_B^{-1}b$
1	$c_B^T - c_B^T A_B^{-1} A_B = 0$	$c_N^T - c_B^T A_B^{-1} A_N$	$-c_B^T A_B^{-1} b$

$\operatorname{minimize}$	$-(x_1+x_2)$
subject to	$x_1 + 2x_2 \le 6$
	$x_1 - x_2 \le 3$
	$x_1, x_2 \ge 0$

	$x_1$	$x_2$	$z_1$	$z_2$	$a_{i0}$
$z_1$	1	2	1	0	6
$z_2$	1	-1	0	1	3
$a_{0j}$	1	1	0	0	0

	$x_1$	$x_2$	$z_1$	$z_2$	$a_{i0}$
$z_1$	0	3	1	-1	3
$x_1$	1	-1	0	1	3
$a_{0j}$	0	2	0	-1	-3

	$x_1$	$x_2$	$z_1$	$z_2$	$a_{i0}$
$x_2$	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
$x_1$	1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
$a_{0j}$	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

# Simple example with cycling

$$\begin{array}{ll} \text{Max} \quad I = 2.3x_1 + 2.15x_2 - 13.55x_3 - 0.4x_4,\\ \text{subject to} & 0.4x_1 + 0.2x_2 - 1.4x_3 - 0.2x_4 \leq 0,\\ & -7.8x_1 - 1.4x_2 + 7.8x_3 + 0.4x_4 \leq 0,\\ & x_i \geq 0, \quad i = 1 \dots 4. \end{array} \tag{1}$$

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	Ι			
0.4	0.2	-1.4	-0.2	1.0			=	0	
-7.8	-1.4	7.8	0.4		1.0		=	0	$T^{(1)}$
-2.3	-2.15	13.55	0.4			1.0	=	0	
1.0	0.5	-3.5	-0.5	2.5			=	0	
	2.5	-19.5	-3.5	19.5	1.0		=	0	$T^{(2)}$
	-1.0	5.5	-0.75	5.75		1.0	=	0	
1.0		0.4	0.2	-1.4	-0.2		=	0	
	1.0	-7.8	-1.4	7.8	0.4		=	0	$T^{(3)}$
		-2.3	-2.15	13.55	0.4	1.0	=	0	

The pivot rule being used is steepest descent: choose the column with the most negative entry in the bottom row.

#### Beale's example of cycling in linear programming.

We are minimizing. After 6 iterations the tableau is the same as at the start.

The pivot column selection rule is steepest descent: choose the column with the most negative entry in the bottom row.

Note that there is degeneracy, in that there are Os in the right hand column.

The value of the objective function stays at 0. However, the true minimum is -1/20.

	x_1	x_2	x_3	×_4	x_5	×_6	x_7	
	$\frac{1}{4}$	-60	$-\frac{1}{25}$	9	1	0	0	0
	$\frac{1}{2}$	-90	$-\frac{1}{50}$	3	0	1	0	0
	0	0	1	0	0	0	1	1
Payoff	- 3/4	150	- <u>1</u> 50	6	0	0	0	0
	<b>1</b>		2					1
	×_1	×_2	×_3 4	X_4	×_5	×_0	×_/	
x_I		-240	25	36	4	U	U	U
	U	30	50	-15	-2	1	U	U
	0	0	1	0	0	0	1	1
Payoff	0	-30	- 750	33	3	0	0	0
	x_1	×_2	x_3	×_4	x_5	×_6	x_7	
x_1	1	0	<u>8</u> 25	-84	-12	8	0	0
×_2	0	1	<u>1</u> 500	- <u>1</u> 2	- <u>1</u> 15	$\frac{1}{30}$	0	0
	0	0	1	0	0	0	1	1
Payoff	0	0	- 2/25	18	1	1	0	0
	x_1	×_2	x_3	×_4	×_5	×_6	x_7	
x_3	<u>25</u> 8	0	1	- 525 2	- 75/2	25	0	0
×_2	- <u>1</u> 160	1	0	$\frac{1}{40}$	<u>1</u> 120	- <u>1</u> 60	0	0
	- <u>25</u> 8	0	0	<u>525</u> 2	<u>75</u> 2	-25	1	1
Payoff	<u>1</u> 4	0	0	-3	-2	3	0	0
	× 1	~ ?	~ 3	~ 1	~ 5	~ 6	~ 7	1
	125	10500	J	4	×_3	150		
x_3	2	10500	1		1	2	U	

	x_1	×_2	x_3	×_4	×_5	×_6	x_7	
×_5	- 5/4	210	$\frac{1}{50}$	0	1	-3	0	0
×_4	$\frac{1}{6}$	-30	- <u>1</u> 150	1	0	$\frac{1}{3}$	0	0
	0	0	1	0	0	0	1	1
Payoff	- 7/4	330	<u>1</u> 50	0	0	-2	0	0

0

0

-50

-1

150

1

1

0

1

0

-10500

120

Payoff

0

0

#### lp\_solve.exe

### lp\_solve.exe [options] "<" <input\_file>

list of	options	
-h		prints this message
-v		verbose mode, gives flow through the program
-d		debug mode, all intermediate results are printed,
		and the branch-and-bound decisions
-р		print the values of the dual variables
-i		print all intermediate valid solutions.
		Can give you useful solutions even if the
		total run time is too long
-t		trace pivot selection

LPO is a file containing lines of:

x1 + x2; row1: x1 + 2 x2 <= 6; row2: x1 - x2 <= 3;

### Mathematica

```
In[127]:= b={6,3}
           c = \{1, 1\}
           m = \{\{1, 2\}, \{1, -1\}\};
AbsoluteTiming[
   For[i=1,i<=100000,i++,
   x=LinearProgramming[-c,-m,-b]];
٦
Print[x];
Out[128] = \{6,3\}
Out[129] = {1,1}
Out[130] = {4.0900000, Null}
           \{4,1\}
```

The problem is solved in about 0.000004 seconds.

The solutionis x1=4, x2=1.

# Simple2x

	* 8 5	N?			
descendenced a	l x 1	× 2	z 1	z 1	T
	1	2	1		c
		2			0
5.55 AM		-1	U	1	E
Payoff	1	1	0	0	0
* Pivot 2,1	M	aximize Ratio	mal Primal	feasible Du	al infeasibl Mar
Simple2x -	lp				
le Tableau	J Simplex	Auto Setti	ngs Pivot	Help	11 - ANG - ANG
		<b>k</b> ?			
	x_1	×_2	z_1	z_1	
	0	3	1	-1	3
x_1	1	-1	0	1	3
×_1 Payoff	1 0	-1 2	0	1 -1	3 -3
x_1 Payoff Pivot 1,2	1 0 M:	-1 2 aximize Ratio	0 0 onal Primal	1 -1 feasible Du	3 -3
x_1 Payoff "Pivot 1,2	1 0	-1 2 aximize Ratio	0 D onal Primal	1 -1 feasible   Du	3 -3 Ial infeasibl Mar
x_1 Payoff Pivot 1,2	1 O Ma	-1 2 aximize Ratio	0 D onal Primal	1 -1 feasible   Du	3 -3 Ial infeasibl Mar
x_1 Payoff Pivot 1,2 Simple2x- ile Tablea	1 0 Ip u Simplex	-1 2 aximize Ratio Auto Setti	0 0 onal Primal	1 -1 feasible   Du Help	3 -3 Ial infeasibl Mar
x_1 Payoff Pivot 1,2 Simple2x- ile Tablea	1 D Ip U Simplex	-1 2 aximize Ratio Auto Setti	0 D onal Primal ings Pivot	1 -1 feasible   Du Help	3 -3 al infeasibl Mar
x_1 Payoff Pivot 1,2 Simple2x- ile Tablea	1 U Ip U Simplex 2 Q @	-1 2 aximize Ratio	0 0 onal Primal ings Pivot z_1	1 -1 feasible Du Help z_1	3 -3 al infeasibl Mar
x_1 Payoff Pivot 1,2 Simple2x- ile Tablea D C D L	1 0 Ip Simplex 2 2 2	-1 2 aximize Ratio Auto Setti k? ×_2 1	0 0 onal Primal ings Pivot z_1 1 3	1 -1 feasible Du Help z_1 - 1/3	3 -3 al infeasibl Mar
x_1 Payoff Pivot 1,2 Simple2x- ile Tablea D C D I	1 0 Ip v Simplex 2 2 2	-1 2 aximize Ratio Auto Setti <u>k?</u> ×_2 1 0	0 0 onal Primal rgs Pivot z_1 1 3 1 3	1 -1 feasible Du Help z_1 - 1 3 2 3	3 -3 al infeasibl Mar

# IBM

Industries & solutions Services

es Products

Support & downloads

My IBM

IBM Software > Products > Business analytics > Prescriptive analytics >

# **CPLEX** Optimizer

High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming



Model business issues mathematically and solve them with IBM ILOG CPLEX Optimizer's powerful algorithms to produce precise and logical decisions.

IBM ILOG CPLEX Optimizer's mathematical programming technology enables decision optimization for improving efficiency, reducing costs, and increasing profitability.

Fundamental algorithms: IBM ILOG CPLEX Optimizer provides flexible, <u>high-performance</u> <u>mathematical programming solvers</u> for <u>linear programming</u>, <u>mixed integer programming</u>, <u>quadratic programming</u>, <u>and quadratically constrained programming</u> problems. These include a distributed parallel algorithm for mixed integer programming to leverage multiple computers to solve difficult problems.

#### **Community Edition**

Problem size limited to 1000 variables and 1000 constraints. All features included. Available on the most popular supported platforms.

## Two phase method

minimize	$y_1 + y_2$	
subject to	$x_1 + x_2 - z_1 + y_1$	= 1
	$2x_1 - x_2 - z_2 + y_2$	= 1
	$3x_2 + z_3$	= 2
	$x_1, x_2, z_1, z_2, z_3, y_1, y_2$	$\geq 0,$

	$x_1$	$x_2$	$z_1$	$z_2$	$z_3$	$y_1$	$y_2$	
$y_1$	1	1	-1	0	0	1	0	1
$y_2$	2	-1	0	-1	0	0	1	1
$z_3$	0	3	0	0	1	0	0	2
II	-6	-3	0	0	0	0	0	0
Ι	3	0	-1	-1	0	0	0	2

	$x_1$	$x_2$	$z_1$	$z_2$	$z_3$	$y_1$	$y_2$	
$y_1$	1	1	-1	0	0	1	0	1
$y_2$	2	-1	0	-1	0	0	1	1
$z_3$	0	3	0	0	1	0	0	2
II	-6	-3	0	0	0	0	0	0
Ι	3	0	-1	-1	0	0	0	2

Phase I now proceeds by pivoting on  $a_{21}$  to get

	$x_1$	$x_2$	$z_1$	$z_2$	$z_3$	$y_1$	$y_2$	
	0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	1	$-\frac{1}{2}$	$\frac{1}{2}$
	1	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$
	0	3	0	0	1	0	0	2
II	0	-6	0	-3	0	0	3	3
Ι	0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	0	$-\frac{3}{2}$	$\frac{1}{2}$

and on  $a_{14}$  to get

	$x_1$	$x_2$	$z_1$	$z_2$	$z_3$	$y_1$	$y_2$	
	0	3	-2	1	0	2	-1	1
	1	1	-1	0	0	1	0	1
	0	3	0	0	1	0	0	2
Π	0	3	-6	0	0	6	0	6
Ι	0	0	0	0	0	-1	-1	0
	<i>m</i> -	<i>m</i> -	~.	~-		~-		
	$\overline{}$	$x_2$	~1	~2		~3		1
	0	3	-2	1		0	1	
	1	1	-1	0		0	1	
	0	3	0	0		1	2	
	0	3	-6	0		0	6	]
								_
	$x_1$	$x_2$	$z_1$	$z_2$		$z_3$		
	0	1	$-\frac{2}{3}$	$\frac{1}{3}$		0	$\frac{1}{3}$	
	1	0	$-\frac{1}{3}$	$-\frac{1}{3}$		0	$\frac{2}{3}$	
	0	0	2	-1		1	1	
	0	0	-4	-1		0	5	

# Dual simplex algorithm

$\operatorname{minimize}$	$2x_1 + 3x_2 + 4x_3$	
subject to	$x_1 + 2x_2 + x_3 - z_1$	= 3
	$2x_1 - x_2 - 3x_3 - z_2$	=4
	$x_1, x_2, x_3, z_1, z_2$	$\geq 0.$

-1	-2	-1	1	0	-3
-2	1	3	0	1	-4
2	3	4	0	0	0

0	$-\frac{5}{2}$	$-\frac{5}{2}$	1	$-\frac{1}{2}$	-1
1	$-\frac{1}{2}$	$-\frac{3}{2}$	0	$-\frac{1}{2}$	2
0	4	7	0	1	-4

0	1	1	$-\frac{2}{5}$	$\frac{1}{5}$	$\frac{2}{5}$
1	0	-1	$-\frac{1}{5}$	$-\frac{2}{5}$	$\frac{11}{5}$
0	0	3	$\frac{8}{5}$	$\frac{1}{5}$	$-\frac{28}{5}$

# Gomory's cutting plane method

0	1	1	$-\frac{2}{5}$	$\frac{1}{5}$	$\frac{2}{5}$
1	0	-1	$-\frac{1}{5}$	$-\frac{2}{5}$	$\frac{11}{5}$
0	0	3	$\frac{8}{5}$	$\frac{1}{5}$	$-\frac{28}{5}$

$$x_2 + x_3 - 1z_1 + 0z_2 \le x_2 + x_3 - \frac{2}{5}z_1 + \frac{1}{5}z_2 = \frac{2}{5},$$

 $x_2 + x_3 - z_1 \le 0.$ 

0	1	1	$-\frac{2}{5}$	$\frac{1}{5}$	0	$\frac{2}{5}$
1	0	-1	$-\frac{1}{5}$	$-\frac{2}{5}$	0	$\frac{11}{5}$
0	0	0	$-\frac{3}{5}$	$-\frac{1}{5}$	1	$-\frac{2}{5}$
0	0	3	$\frac{8}{5}$	$\frac{1}{5}$	0	$-\frac{28}{5}$
0	1	1	-1	0	1	0
1	0	-1	1	0	-2	3
0	0	0	3	1	-5	2
0	0	9	1	0	1	6

#### Markov Chains, Computer Proofs, and Average-Case Analysis of Best Fit Bin Packing

E.G. Coffman, Jr.<sup>1</sup>, D. S. Johnson<sup>1</sup>, P. W. Shor<sup>1</sup>, R. R. Weber<sup>2</sup>

Abstract. Many complex processes can be modeled by (countably) infinite, multi-dimensional Markov chains. Unfortunately, current theoretical techniques for analyzing infinite Markov chains are for the most part limited to three or fewer dimensions. In this paper we propose a computer-aided approach to the analysis of higher-dimensional domains, using several open problems about the average-case behavior of the Best Fit bin packing algorithm as case studies. We show how to use dynamic and linear programming to construct potential functions that, when applied to suitably modified multi-step versions of our original Markov chain, yield drifts that are bounded away from 0. This enables us to completely classify the expected behavior of Best Fit under discrete uniform distributions  $U\{J,K\}$  when K is small. (Under  $U\{J,K\}$ , the allowed item sizes are i/K,  $1 \le i \le J$ , with all J possibilities equally likely.) In addition, we can answer yes to the long-standing open question of whether there exist distributions of this form for which Best Fit yields linearly-growing waste. The proof of the latter theorem relies on a 24-hour computation, and although its validity does not depend on the linear programming package we used, it does rely on the correctness of our dynamic programming code and of our computer's implementation of the IEEE floating point standard.

### Markov Chains, Computer Proofs, and Average-Case Analysis of Best Fit Bin Packing

E.G. Coffman, Jr.<sup>1</sup>, D. S. Johnson<sup>1</sup>, P. W. Shor<sup>1</sup>, R. R. Weber<sup>2</sup>

Many complex processes can be modelled by (countably) infinite, multidimensional Markov chains. Unfortunately, theoretical techniques for analysing infinite Markov chains are for the most part limited to three or fewer dimensions. In this paper we propose a computer-aided approach to the analysis of higher-dimensional domains, using several open problems about the average-case behavior of the Best Fit bin packing algorithm as case studies. We show how to use dynamic and lienar programming to construct potential functions that when applied to suitably modified multi-step versions of our original Markov chain, yield drifts that are bounded away from 0. This enables us to completely classify the expected behavior of Best Fit under discrete uniform distributions U{J, K} when K is small. In addition, we can answer yes to the long-standing open question of whether there exist distributions of this form for which Best Fit yields linearly-growing waste. The proof of the latter theorem relies on a 24-hour computation, and although its validity does not depend on the linear programming package we used, it does rely on the correctness of our dynamic programming code and of our computer's implementation of the IEEE floating point standard.

These results are based on the use of a particular kind of potential function (called a *Lyapunov* function in the literature). Once one goes beyond three dimensions, there are no longer simple constructions for the needed functions. There are, however, algorithmic approaches to determining whether Lyapunov functions of certain standard types exist, and the remainder of Section 2 describes them. We show how linear programming can be used to test for the existence of both linear and quadratic Lyapunov functions, using an old lemma of Hajek [9] and a new one of our own.

> Maximize  $\gamma$ , subject to  $\sum_{i=1}^{d} a_i \Delta_i(s) < -\gamma, \text{ for all } s \in S_{k,i}$   $a_i \ge 0, \ 1 \le i \le d, \text{ and}$   $\sum_{i=1}^{d} a_i \le 1.$

#### Example: On-line bin packing

An infinite sequence of items are to be packed into bins of size 5. Each successive item is equally likely to be of sizes 1, 2 or 3. One possible packing algorithm is *Best Fit* (BF), which puts each item into the smallest gap into which it will fit amongst the existing gaps in partially full bins, or if there is no gap large enough, the item goes into an empty bin.

The state at time t is written  $x = (x_1, x_2, x_3, x_4)^T$ where  $x_i$  is the number of partially full bins with a gap of size *i*. Then, for example,

Let d(x) = E[x(t+1) - x(t) | x(t) = x], be the expected drift, e.g.,  $d(0, 4, 0, 0) = \left(\frac{1}{3}, -\frac{1}{3}, 0, 0\right)^T$ .

**Problem:** Let  $E|x(t)| = x_1 + x_2 + x_3 + x_4$  be the number of partially full bins. Does  $E|x(t)| \to \infty$  or is E|x(t)| bounded as  $t \to \infty$ ?

Practical considerations suggest that it is better if E|x(t)| is bounded than if  $E|x(t)| \to \infty$ .

Theorem E|x(t)| is bounded as  $t \to \infty \iff \exists \delta < 0$  and potential function  $\phi(x) \ge 0$  such that  $E[\phi(x(t+1)) - \phi(x) \mid x(t) = x] \le \delta < 0$  for all x. This theorem makes series because it gives that from

This theorem makes sense because it says that from every possible starting state the value of the potential function is drifting (on average) towards 0.

Suppose we try  $\phi(x) = \sum_{i=1}^{4} a_i x_i$  and hunt for  $a_i$  that will work by considering the LP

minimize  $\delta$ 

subject to 
$$d(x)^T a \leq \delta$$
, for all  $x$   
 $a_1, a_2, a_3, a_4 \geq 0$ .

If the optimal solution is  $\delta < 0$  this proves that E|x(t)| is bounded as  $t \to \infty$ .

This is what happens for our example. The LP has only 6 constraints.

For packing items of sizes 1, ..., 8 into bins of size 14 a version of this method gives a LP with 415,953 constraints and takes 24 hours to construct and solve!

For packing  $1, \ldots, 8$  into bins of size 11 the answer can be shown to be  $E|x(t)| \to \infty$ .

	K	5	6	7	8	9	10	11	12	13	14
1	3	B-L2	B-L2	B-L2	B-Q1	B-Th	B-Th	B-Th	B-Th	B-Th	B-Th
	4		B-L4	B-L3	B-Q1	B-Q1	B-Q1	B-Q1	B-Q1	B-Q1	B-Th
1	5			B-L23	B-Q1	B-Q1	B-Q1	<b>B-Q1</b>	B-Q1	B-Q1	B-Q1
	6				B-Q2	B-Q5	B-Q1	<b>B-Q1</b>	B-Q1	<b>B-Q1</b>	<b>B-Q1</b>
1	7	ĺ				B-Q7	B-Q15	B-Q2	B-Q2	B-Q1	B-Q1
	8						B-Q13	Ln-P	B-x	B-x	B-Q7
1	9							B-x	Ln-P	Ln-x	B-x
ļ	10								B-x	Ln-x	Ln-x
1	11									B-x	Ln-x
	12										B-x

**TABLE 2.** Results proved for  $E[w_{BF}(L_{N,J,K})]$ .

Section 3 then uses this technology to prove bounded expected waste for many of the entries in Table 1. A proof in this case starts with the construction by computer of a linear program that, although it has few variables, may sometimes have more than 100,000 constraints. The linear program is then solved using a standard LP package. The package we use is CPLEX<sup>™</sup> (CPLEX is a trademark of CPLEX Optimization, Inc.), but the validity of our proofs is independent of the correctness of CPLEX, since after CPLEX generates a solution, we verify the validity of that solution using our own code. The correctness of our proofs does, however, depend on the correctness of our generation and checking programs (for which we shall present listings of key routines in the final paper), and on the fact that our computer runs properly and correctly implements the IEEE floating point standard [1].

# The running time of algorithms



Suppose instances of size 1000 can be solved in one second.

Technology increases computing speed by a factor of 16.

What size of problem can we now solve in one second?

If  $T(n) = n^2$  we can now solve problems of size 4000.

If  $T(n) = 2^n$  we can now solve problems of size 1004.

# Relationship between complexity classes







# **Clay Mathematics Institute**

Dedicated to increasing and disseminating mathematical knowledge

ABOUT PROGRAMS

MILLENNIUM PROBLEMS

PEOPLE PL

PUBLICATIONS

 ${\sf P} \; {\sf v} \; {\sf NP}$  Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

## A Turing machine

The machine has states A and B. In each state, the machine reads the bit under the head and executes the instructions in the following table (where Pn prints bit n, L means "move left" and R means "move right, and A and B mean "switch to that state").

	А	В
0	$\mathrm{P1},\mathrm{R},\mathrm{B}$	P2, L, A
1	P2, L, A	P2, R, B
2	P1, L, A	$\mathrm{P0},\mathrm{R},\mathrm{A}$

				А						
 0	0	0	0	0	0	0	0	0	0	
					В					
 0	0	0	0	1	0	0	0	0	0	
				А						
 0	0	0	0	1	2	0	0	0	0	
			А							
 0	0	0	0	2	2	0	0	0	0	

# Karp's 21 NP-complete problems

From Wikipedia, the free encyclopedia

• **Satisfiability**: the boolean satisfiability problem for formulas in conjunctive normal form (SAT)

- **0–1 integer programming** (A variation in which only the restrictions must be satisfied, with no optimization)
- **Clique** (independent set problem)
  - Set packing
  - Vertex cover
    - Set covering
    - Feedback node set
    - Feedback arc set
    - Directed Hamilton circuit
      - Undirected Hamilton circuit
- Satisfiability with at most 3 literals per clause (equivalent to 3-SAT)
  - Chromatic number (also called the Graph Coloring Problem)
    - Clique cover
    - Exact cover
      - Hitting set
      - Steiner tree
      - 3-dimensional matching
      - Knapsack (Karp's definition of Knapsack is closer to Subset sum)
        - Job sequencing
        - Partition
          - Max cut

# Firing squad problem

# Firing Squad Synchronization, Computer Science's Most Macabre-Sounding Problem

Written by BEN RICHMOND

July 14, 2015 // 12:00 PM EST

The name of the problem comes from an analogy with real-world firing squads: the goal is to design a system of rules according to which an officer can so command an execution detail to fire that its members fire their rifles simultaneously.

More formally, the problem concerns cellular automata, arrays of finite state machines called "cells" arranged in a line, such that at each time step each machine transitions to a new state as a function of its previous state and the states of its two neighbors in the line. For the firing squad problem, the line consists of a finite number of cells, and the rule according to which each machine transitions to the next state should be the same for all of the cells interior to the line, but the transition functions of the two endpoints of the line are allowed to differ, as these two cells are each missing a neighbor on one of their two sides.

The states of each cell include three distinguished states: "active", "quiescent", and "firing", and the transition function must be such that a cell that is quiescent and whose neighbors are quiescent remains quiescent. Initially, at time t = 0, all states are quiescent except for the cell at the far left (the general), which is active. The goal is to design a set of states and a transition function such that, no matter how long the line of cells is, there exists a time t such that every cell transitions to the firing state at time t, and such that no cell belongs to the firing state prior to time t.

## Klee and Minty polytope



 $\epsilon \le x_1 \le 1,$  $\epsilon x_{i-1} \le x_i \le 1 - \epsilon x_{i-1} \quad \text{for } i = 2, \dots, n$ 

# The Hirsch conjecture



 $\Delta(n,m) =$ maximum distance between two vertices of a polytope in  $\mathbb{R}^n$  that is defined by m inequalities.

#### Hirsch Conjecture: $\Delta(n,m) \leq m-n$ .

A counterexample to the Hirsch Conjecture was discovered by Francisco Santos in 2010. However, it remains an open question whether or not  $\Delta(n, m) =$  is bounded by some polynomial function of m and n.

## A counterexample to the Hirsch conjecture

Francisco Santos<sup>\*</sup>

To Victor L. Klee (1925–2007), in memoriam<sup>†</sup>

#### Abstract

The Hirsch Conjecture (1957) stated that the graph of a *d*-dimensional polytope with *n* facets cannot have (combinatorial) diameter greater than n-d. That is, that any two vertices of the polytope can be connected to each other by a path of at most n-d edges.

This paper presents the first counterexample to the conjecture. Our polytope has dimension 43 and 86 facets. It is obtained from a 5-dimensional polytope with 48 facets which violates a certain generalization of the *d*step conjecture of Klee and Walkup.

#### 1 Introduction

The Hirsch conjecture is the following very fundamental statement about the combinatorics of polytopes. It was stated by Warren M. Hirsch in 1957 in the context of the *simplex method*, and publicized by G. Dantzig in his 1963 monograph on linear programming [10]:

The (combinatorial) diameter of a polytope of dimension d with n facets cannot be greater than n - d.

Here we call *combinatorial diameter* of a polytope the maximum number of steps needed to go from one vertex to another, where a step consists in traversing an edge. Since we never refer to any other diameter in this paper, we will often omit the word "combinatorial".

Our main result is the construction of a 43-dimensional polytope with 86 facets and diameter (at least) 44. Via products and glueing copies of it we can also construct an infinite family of polytopes in fixed dimension d with increasing number n of facets and of diameter  $(1 + \epsilon)(n - d)$ , for a positive constant  $\epsilon$ .

### The ellipsoid method



Figure 6: A step of the ellipsoid method where  $x_t \notin P$  but  $x_{t+1} \in P$ . The polytope P and the half-ellipsoid that contains it are shaded.

**Theorem 6.2.** Let E = E(z, D) be an ellipsoid in  $\mathbb{R}^n$  and  $a \in \mathbb{R}^n$  non-zero. Consider the half-space  $H = \{x \in \mathbb{R}^n : a^T x \ge a^T z\}$ , and let

$$z' = z + \frac{1}{n+1} \frac{Da}{\sqrt{a^T Da}},$$
$$D' = \frac{n^2}{n^2 - 1} \left( D - \frac{2}{n+1} \frac{Daa^T D}{a^T Da} \right)$$

Then D' is symmetric and positive definite, and therefore E' = E(z', D') is an ellipsoid. Moreover,  $E \cap H \subseteq E'$  and  $\operatorname{Vol}(E') < e^{-1/(2(n+1))} \operatorname{Vol}(E)$ . Proof sketch of Theorem 6.2. We prove the theorem for  $E = \{x \in \mathbb{R}^n : x^T x \leq 1\}$  and  $H = \{x \in \mathbb{R}^n : x_1 \geq 0\}$ . Since every pair of an ellipsoid and a hyperplane as in the statement of the theorem can be obtained from E and H via some affine transformation, the general case then follows by observing that affine transformations preserve inclusion and, by Lemma 6.3, relative volume of sets.

Let  $e_1 = (1, 0, \dots, 0)^T$ . Then,

$$E' = E\left(\frac{e_1}{n+1}, \frac{n^2}{n^2 - 1}\left(I - \frac{2}{n+1}e_1e_1^T\right)\right)$$
$$= \left\{x \in \mathbb{R}^n : \frac{n^2 - 1}{n^2}\sum_{i=1}^n x_i^2 + \frac{1}{n^2} + \frac{2(n+1)}{n^2}x_1(x_1 - 1) \le 1\right\}.$$

Consider an arbitrary  $x \in E \cap H$ , and observe that  $0 \le x_1 \le 1$  and  $\sum_{i=1}^n x_i^2 \le 1$ . It is easily verified that  $x \in E'$  and thus  $E \cap H \subseteq E'$ .

Now consider the affine transformation  $F : \mathbb{R}^n \to \mathbb{R}^n$  given by

$$F(x) = \frac{e_1}{n+1} + \left(\frac{n^2}{n^2 - 1} \left(I - \frac{2}{n+1}e_1e_1^T\right)\right)^{\frac{1}{2}}x.$$

It is not hard to show that E' = F(E). Therefore, by Lemma 6.3,

$$\begin{aligned} \frac{\operatorname{Vol}(E')}{\operatorname{Vol}(E)} &= \sqrt{\operatorname{det}\left(\frac{n^2}{n^2 - 1}\left(I - \frac{2}{n+1}e_1e_1^T\right)\right)} \\ &= \left(\frac{n^2}{n^2 - 1}\right)^{\frac{n}{2}}\left(1 - \frac{2}{n+1}\right)^{\frac{1}{2}} = \frac{n}{n+1}\left(\frac{n^2}{n^2 - 1}\right)^{\frac{n-1}{2}} \\ &= \left(1 - \frac{1}{n+1}\right)\left(1 + \frac{1}{n^2 - 1}\right)^{\frac{n-1}{2}} < e^{-\frac{1}{n+1}}\left(e^{\frac{1}{n^2 - 1}}\right)^{\frac{n-1}{2}} = e^{-\frac{1}{2(n+1)}}, \end{aligned}$$

where the strict inequality follows by using twice that  $1 + a < e^a$  for all  $a \neq 0$ .

### Ellipsoid method

#### 7.1 Ellipsoid method

Consider a polytope  $P = \{x \in \mathbb{R}^n : Ax \ge b\}$ , given by a matrix  $A \in \mathbb{Z}^{m \times n}$  and a vector  $b \in \mathbb{Z}^m$ . Assume for now that P is bounded and either empty or full-dimensional. Here, P is called **full-dimensional** if Vol(P) > 0. The ellipsoid method takes the following steps to decide whether P is non-empty:

- 1. Let U be the largest absolute value among the entries of A and b, and define  $x_0 = 0$ ,  $D_0 = n(nU)^{2n}I$ ,  $E_0 = E(x_0, D_0)$ ,  $V = (2\sqrt{n})^n (nU)^{n^2}$ ,  $v = n^{-n} (nU)^{-n^2(n+1)}$ ,  $t^* = \lceil 2(n+1) \log(V/v) \rceil$ .
- 2. For  $t = 0, \ldots, t^*$ , do the following:
  - 1. If  $t = t^*$  then stop; P is empty.
  - 2. If  $x_t \in P$  then stop; P is non-empty.
  - 3. Find a violated constraint, i.e. a row j such that  $a_j^T x_t < b_j$ .
  - 4. Let  $E_{t+1} = E(x_{t+1}, D_{t+1})$  with

$$x_{t+1} = x_t + \frac{1}{n+1} \frac{D_t a_j}{\sqrt{a_j^T D_t a_j}},$$
$$D_{t+1} = \frac{n^2}{n^2 - 1} \left( D_t - \frac{2}{n+1} \frac{D_t a_j a_j^T D_t}{a_j^T D_t a_j} \right).$$
# Karmarkar's algorithm

From Wikipedia, the free encyclopedia

Karmarkar's algorithm is an algorithm introduced by Narendra Karmarkar in 1984 for solving linear programming problems. It was the first reasonably efficient algorithm that solves these problems in polynomial time. The ellipsoid method is also polynomial time but proved to be inefficient in practice.

Where n is the number of variables and L is the number of bits of input to the algorithm, Karmarkar's algorithm requires  $O(n^{3.5}L)$  operations on O(L) digit numbers, as compared to  $O(n^6L)$  such operations for the ellipsoid algorithm. The runtime of Karmarkar's algorithm is thus

 $O(n^{3.5}L^2 \cdot \log L \cdot \log \log L)$ 

using FFT-based multiplication (see Big O notation).

Karmarkar's algorithm falls within the class of interior point methods: the current guess for the solution does not follow the boundary of the feasible set as in the simplex method, but it moves through the interior of the feasible region, improving the approximation of the optimal solution by a definite fraction with every iteration, and converging to an optimal solution with rational data.<sup>[1]</sup>

#### Patent controversy - Can Mathematics be patented? [edit]

At the time he invented the algorithm, Narendra Karmarkar was employed by AT&T. After applying the algorithm to optimizing AT&T 's telephone network,<sup>[14]</sup> they realized that his invention could be of practical importance. In April 1985, AT&T promptly applied for a patent on Karmarkar's algorithm and that became more fuel for the ongoing controversy over the issue of software patents.<sup>[15]</sup> This left many mathematicians uneasy, such as Ronald Rivest (himself one of the holders of the patent on the RSA algorithm), who expressed the opinion that research proceeded on the basis that algorithms should be free. Even before the patent was actually granted, some claimed that there might have been prior art that was applicable.<sup>[16]</sup>

Mathematicians who specialize in numerical analysis such as Philip Gill and others claimed that Karmarkar's algorithm is equivalent to a projected Newton barrier method with a logarithmic barrier function, if the parameters are chosen suitably.<sup>[17]</sup> However, Gill's argument is flawed, insofar as the method they describe does not even qualify as an "algorithm", since it requires choices of parameters that don't follow from the internal logic of the method, but rely on external guidance, essentially from Karmarkar's algorithm.<sup>[18]</sup> Furthermore, Karmarkar's contributions are considered far from obvious in light of all prior work, including Fiacco-McCormick, Gill and others cited by Saltzman.<sup>[18][19][20]</sup> The patent was debated in the U.S. Senate and granted in recognition of the essential originality of Karmarkar's work, as U.S. Patent 4,744,026 🖙: "Methods and apparatus for efficient resource allocation" in May 1988. AT&T supplied the KORBX system<sup>[21] [22]</sup> based on this patent to the Pentagon,<sup>[23][24]</sup> which enabled them to solve mathematical programming problems which were previously considered unsolvable.

Opponents of software patents have further alleged that the patents ruined the positive interaction cycles that previously characterized the relationship between researchers in linear programming and industry, and specifically it isolated Karmarkar himself from the network of mathematical researchers in his field. <sup>[25]</sup>

The patent itself expired in April 2006, and the algorithm is presently in the public domain.

# A basic feasible flow is a spanning tree







#### Network simplex algorithm



The demand for electricity at node *i* is  $d_i$ . Node *i* has  $k_i$  generators, that can generate electricity at costs of  $a_{i1}, \ldots, a_{ik_i}$ , up to amounts  $b_{i1}, \ldots, b_{ik_i}$ . There are n = 12 nodes and 351 generators in all. The capacity for transmission from node *i* to *j* is  $c_{ij}$  (=  $c_{ji}$ ).

Let  $x_{ij}$  = amount of electricity carried  $i \rightarrow j$  and let  $y_{ij}$  = amount of electricity generated by generator j at node i. The LP is

minimize 
$$\sum_{ij} a_{ij} y_{ij}$$
subject to 
$$\sum_{j} y_{ij} - \sum_{j} x_{ij} + \sum_{j} x_{ji} = d_i, \quad i = 1, \dots, 12,$$
$$0 \le x_{ij} \le c_{ij}, \ 0 \le y_{ij} \le b_{ij}.$$

In addition, there are constraints on the maximum amount of power that may be shipped across the cuts shown by the dotted lines in the diagram.

#### Longest path problem

A set of jobs is to be processed subject to precedence constraints. Job *i* has a duration  $\tau_i$ . What is the least time in which all jobs can be completed?

Consider a graph in which there is an edge (i, j) whenever job *i* must be completed before job *j*. Introduce two additional jobs, *s* and *s'*, each of zero duration, to indicate the start and finish of the project, and introduce edges (s, i) and (i, s') for every job *i*. Suppose we start job *i* at time  $t_i$ . We wish to

minimize  $(t_{s'} - t_s)$ , subject to  $t_j - t_i \ge \tau_i$ , for all  $(i, j) \in A$ .



The dual of this problem is

maximize 
$$\sum_{(i,j)\in A} \tau_i f_{ij}$$

subject to

$$\sum_{j: (j,i) \in A} f_{ji} - \sum_{j: (i,j) \in A} f_{ij} = -b_i, \text{ for all } i, \text{ and } f_{ij} \ge 0, \text{ for all } (i,j) \in A,$$

where  $b_s = 1$ ,  $b_{s'} = -1$  and  $b_i = 0$  for  $i \neq s, s'$ . This is a minimum cost flow problem with  $c_{ij} = -\tau_i$ . The path of edges for which  $f_{ij} = 1$  defines the **critical path**.





Flow network for the assignment problem and solution by the Hungarian algorithm



Hungarian Algorithm for the assignment problem





Put node number 0 on the right, and maximal on the left.

Edges where c\_ij = lambda\_ilambda\_j are shown red.

Decrease as far a possible the node numbers on the right.



Find a maximal matching in the equality graph.. (Shown bold red).





We increase by 1 the node numbers on vertices of S (shaded).

Notice that one edge enters the equality graph; another leaves. But the matching still remains.



There is an augmenting path and the matching increases by 1.

We are done.









# Bellman-Ford algorithm









(c)



5 t х 2 6 8 s 0 7 9 v (e)

### Dijkstra's algorithm























Minimum spanning trees



Minimum spanning tree on US airports





Theorem 11.2. Let (V,E) be a graph with edge costs cij for all (i,j)  $\in$  E. Let U  $\subseteq$  V and (u,v)  $\in$  U × (V \ U) such that cuv = min (i,j) $\in$ U×(V \U) c ij . Then there exists a spanning tree of minimum cost that contains (u,v).

## Prim's algorithm



In this example, Prim's algorithm adds edges in the sequence {1,3}, {3,6}, {6,4}, {3,2}, {2,5}.

# Interior-point method in operation



#### Dakin's method







Figure 16: Illustration of Dakin's method, applied to the IP of Example 13.1

- Start with  $U = \infty$  and  $L = \{X\}$ . By solving the LP relaxation for X, we find that  $\ell(X) = \min_{x \in \tilde{X}} f(x) = f(x^0) = -7/2$  for  $x^0 = (3/2, 5/2)$ .
- Set X is the only candidate for branching and can be split into  $X^1 = \mathbb{Z}^2 \cap \tilde{X}^1$ and  $X^2 = \mathbb{Z}^2 \cap \tilde{X}^2$ , where

 $\tilde{X}^1 = \{ x \in \tilde{X} : x_2 \le 2 \}$  and  $\tilde{X}^2 = \{ x \in \tilde{X} : x_2 \ge 3 \}.$ 

- Bound  $X^1$  and  $X^2$  by solving the corresponding LP relaxations, and obtain  $\ell(X^1) = \min_{x \in \tilde{X}^1} f(x) = f(x^1) = -13/4$  for  $x^1 = (3/4, 2)$ , and  $\tilde{X}^2 = \emptyset$ . We thus set  $L = \{X^1\}$ .
- Branch by splitting  $X^1$  into  $X^3 = \mathbb{Z}^2 \cap \tilde{X}^3$  and  $X^4 = \mathbb{Z}^2 \cap \tilde{X}^4$ , where

$$\tilde{X}^3 = \{ x \in \tilde{X}^1 : x_1 \le 0 \}$$
 and  $\tilde{X}^4 = \{ x \in \tilde{X}^1 : x_1 \ge 1 \},\$ 

• Bound  $X^3$  and  $X^4$  to obtain  $\ell(X^3) = \min_{x \in \tilde{X}^3} f(x) = f(x^3) = -3$  for  $x^3 = (0, 3/2)$  and  $\ell(X^4) = \min_{x \in \tilde{X}^4} f(x) = f(x^4) = -3$  for  $x^4 = (1, 2)$ . Since  $x^4 \in X$ , we set  $U = f(x^4) = -3$ . Then,  $\ell(X^3) \ge U$ , so we can discard  $X^3$  and are done.



Figure 17: Search tree explored by Dakin's method for the IP of Example 13.1

# Neighbourhood search is defeated by local minima

$$A = \begin{pmatrix} 0 & 1 & 0 & 4 & 4 \\ 4 & 0 & 1 & 0 & 4 \\ 4 & 4 & 0 & 1 & 0 \\ 0 & 4 & 4 & 0 & 1 \\ 1 & 0 & 4 & 4 & 0 \end{pmatrix}.$$

There are 4! = 24 TSP tours, and

c(12345) = 5,	c(13245) = 6,	c(14235) = 10,	c(15234) = 6,
c(12354) = 6,	c(13254) = 12,	c(14253) = 20,	c(15243) = 12,
c(12435) = 6,	c(13425) = 10,	c(14325) = 17,	c(15324) = 12,
c(12453) = 10,	c(13452) = 6,	c(14352) = 9,	c(15342) = 17,
c(12534) = 10,	c(13524) = 0,	c(14523) = 10,	c(15423) = 17,
c(12543) = 17,	c(13542) = 12,	c(14532) = 17,	c(15432) = 20.

It is easily verified that the tour 12345 is a local optimum under the 2-OPT neighborhood, while the global optimum is the tour 13524.

#### Tour construction heuristics



Farthest insertion heuristic



The largest solved traveling salesman problem, an 85,900-city route calculated in 2006. The layout of the "cities" corresponds to the design of a customized computer chip created at Bell Laboratories, and the solution exhibits the shortest path for a laser to follow as it sculpts the chip.(*Illustration: Courtesy of David Applegate, Robert Bixby, Vasek Chvatal and William Cook*)



One Fish Two Fish Red Fish Black Fish" is a continuous line drawing constructed from the solution to a 1500-city instance of the Traveling Salesman Problem. The line is a simple closed curve drawn with white ink. It divides the plane into two regions: *in* (drawn with red ink) and *out*(drawn with black). From afar, the piece looks like an alternating link, a knot formed from two interlaced loops, one red and one black.

On April 20th, 2001, David Applegate, Robert Bixby, Vašek Chvátal, and William Cook announced the solution of a traveling salesman problem through 15,112 cities in Germany. This was the largest TSPLIB instance that had been solved at the time, exceeding the 13,509-city tour through the United States that was solved in 1998.

The computation was carried out on a network of 110 processors located at Rice University and at Princeton University. The total computer time used in the computation was 22.6 years, scaled to a Compaq EV6 Alpha processor running at 500 MHz.



n May 2004, the traveling salesman problem of visiting all 24,978 cities in Sweden was solved: a tour of length 855,597 <u>TSPLIB</u> units (approximately 72,500 kilometers) was found and it was proven that no shorter tour exists. At the time of the computation, this was the largest solved TSP instance, surpassing the previous record of <u>15,112</u> <u>cities</u> through Germany set in April 2001.



#### Lemke-Howson for a symmetric game



	$x_1$	$x_2$	$x_3$	$z_1$	$z_2$	$z_3$
$v_0$ :	0	0	0	1	1	1
$v_1$ :	0	$\frac{1}{3}$	0	0	1	$\frac{1}{3}$
$v_2$ :	$\frac{1}{6}$	$\frac{1}{3}$	0	0	1	Ŏ
$v_3$ :	0	$\frac{1}{3}$	$\frac{1}{6}$	0	$\frac{1}{2}$	0
$v_4$ :	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	0	0

#### **Complementary pivoting**

Consider for example the bimatrix game given by

$$P = \begin{pmatrix} 3 & 3 \\ 2 & 5 \\ 0 & 6 \end{pmatrix} \quad \text{and} \quad Q = \begin{pmatrix} 3 & 2 \\ 2 & 6 \\ 3 & 1 \end{pmatrix},$$

Indexing z and w by  $M = \{1, ..., m\}$  and  $N = \{m + 1, ..., m + n\}$ , respectively, the constraint  $Q^T x + w = 1$  can be written in tableau form as follows:

$x_1$	$x_2$	$x_3$	$w_4$	$w_5$	
3	2	3	1	0	1
2	6	1	0	1	1

Assume that label  $\ell = 2$  is dropped by increasing  $x_2$  from 0. By pivoting we obtain the following tableau:

$\frac{7}{3}$	0	83	1	$-\frac{1}{3}$	$\frac{2}{3}$
$\frac{1}{3}$	1	$\frac{1}{6}$	0	$\frac{1}{6}$	$\frac{1}{6}$

The second row now corresponds to variable  $x_2$  that has entered the basis. On the other hand, variable  $w_5$  has left the basis. We thus want to turn to the constraint Py + z = 1 and drop the duplicate label 5. The initial tableau for this constraint looks as follows:

	$y_4$	$y_5$	$z_1$	$z_2$	$z_3$	
Γ	3	3	1	0	0	1
	<b>2</b>	5	0	1	0	1
	0	6	0	0	1	1

$x_1$	$x_2$	$x_3$	$w_4$	$w_5$	
$\frac{7}{3}$	0	83	1	$-\frac{1}{3}$	$\frac{2}{3}$
$\frac{1}{3}$	1	$\frac{1}{6}$	0	$\frac{1}{6}$	$\frac{1}{6}$
$y_4$	$y_5$	$z_1$	$z_2$	$z_3$	
$\frac{y_4}{3}$	$\frac{y_5}{3}$	$\frac{z_1}{1}$	$\frac{z_2}{0}$	$\frac{z_3}{0}$	1
$\begin{array}{c} y_4 \\ 3 \\ 2 \end{array}$	$\frac{y_5}{3}$	$\frac{z_1}{1}$ 0	$\frac{z_2}{0}$	$\frac{z_3}{0}$	1 1

By pivoting on the second column, corresponding to  $y_5$ , and on the third row, we pick up label 3 and obtain the following tableau:

$y_4$	$y_5$	$z_1$	$z_2$	$z_3$	
3	0	1	0	$-\frac{1}{2}$	$\frac{1}{2}$
2	0	0	1	$-\frac{5}{6}$	$\frac{1}{6}$
0	1	0	0	$\frac{1}{6}$	$\frac{1}{6}$

Pivoting one more time in each of the two polytopes, we drop label 3 to pick up label 4:

$x_1$	$x_2$	$x_3$	$w_4$	$w_5$	
$\frac{7}{8}$	0	1	<u>3</u> 8	$-\frac{1}{8}$	$\frac{1}{4}$
$\frac{3}{16}$	1	0	$-\frac{3}{48}$	$\frac{3}{16}$	$\frac{1}{8}$

and then drop label 4 to pick up label 2:

$y_4$	$y_5$	$z_1$	$z_2$	$z_3$	
0	0	1	$-\frac{3}{2}$	$\frac{3}{4}$	$\frac{1}{4}$
1	0	0	$\frac{1}{2}$	$-\frac{5}{12}$	$\frac{1}{12}$
0	1	0	0	$\frac{1}{6}$	$\frac{1}{6}$

At this point we have a fully labelled pair. The final tableaus are the final two above. Reading off the values of x and y from the last column of each tableau and scaling them appropriately yields the equilibrium x = (0, 1/3, 2/3), y = (1/3, 2/3).

# Sperner's lemma



# Sperner's lemma



#### A saddle point



 $\max_{x \in X} \min_{y \in Y} p(x, y) = \min_{y \in Y} \max_{x \in X} p(x, y)$ 

Marquis de Condorcet's paradox, 1786

voter	1	2	3	_
	a	b	с	
	b	с	a	
	с	a	b	

Preferences of three types of voters over three alternatives and a graph of majority relations.



At the top of each column of the table is the number of voters having the preference order below.

There is a directed edge in the graph from alternative x to alternative y indicates when a majority of the voters prefer x to y.
Lagniappe to Journal of Political Economy, 109(6), December 2001.

## An Early Example of the "Winner's Curse" in an Auction

After murdering the Roman Emperor Pertinax (March 28, A.D.193), the praetorian guards had an empire to dispose of:

Then ensued a most disgraceful business and one unworthy of Rome. For, just as if it had been in some market or auction-room, both the City and its entire empire were auctioned off. The sellers were the ones who had slain their emperor, and the would-be buyers were Sulpicianus and Julianus, who vied to outbid each other, one from the inside, the other from the outside [of the praetorian camp]. They gradually raised their bids up to twenty thousand sesterces per soldier. Some of the soldiers would carry word to Julianus, "Sulpicianus offers so much; how much more do you make it?" And to Sulpicianus in turn, "Julianus promises so much; how much do you raise him?" Sulpicianus would have won the day, being inside and being prefect of the city and also the first to name the figure twenty thousand, had not Julianus raised his bid no longer by a small amount but by five thousand at one time, both shouting it in a loud voice and also indicating the amount with his fingers. So the soldiers, captivated by this excessive bid and at the same time fearing that Sulpicianus might avenge Pertinax (an idea that Julianus put into their heads), received Julianus inside and declared him emperor. [Dio's Roman History, book LXXIV]

Auction theorists will note the success of the final "jump" bid in what we would now call an English auction. Assuming Julianus committed to pay all of the more than 10,000 praetorians, the total bid corresponded to well over a billion dollars today. (We approximated the current value by reference to the salaries of elite soldiers. The winning bid corresponded to about five years' wages per praetorian.) We know of many smaller auctions in the early Roman Empire, a loan of seven million of sesterces (when prices were somewhat lower than in 193), and imperial revenues that must have been far larger, but this bid appears enormous even for the fully monetized market economy of the early Empire. Julianus appears to have defaulted on his bid in any case, and the soldiers' resulting anger may have contributed to his short tenure in office.

Two months later, Septimius Severus marched an army to Rome, the praetorians deserted Julianus, and he was murdered after a reign of only 66 days. In Gibbon's words, Julianus was "beheaded as a common criminal, after having purchased, with an immense treasure, an anxious and precarious reign of only sixty-six days." Severus then disbanded the troublesome praetorians.

[Dio's Roman History, book LXXIV, translated by Earnest Cary (Cambridge, Mass.: Harvard Univ. Press, 1960; Edward Gibbon, The Decline and Fall of the Roman Empire (1776), vol. 1, chap. V.]

Suggested by Paul Klemperer and Peter Temin