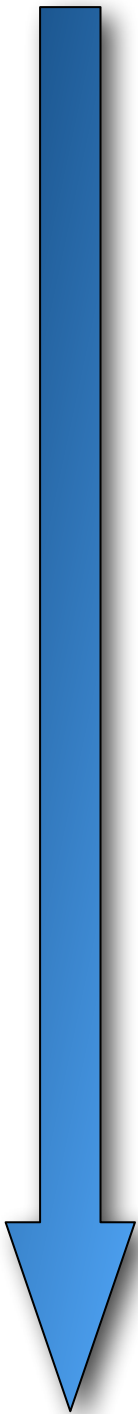


Linear Optimization: A mathematical success story



Jean Baptiste Joseph Fourier
(1768—1830)



Fourier-Motzkin elimination
for solving system of
linear inequalities (1827)

Leonid Vitaliyevich Kantorovich
(1912—1986)

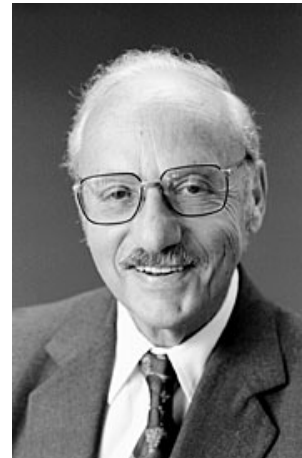


developed earliest
linear optimisation models
during World War II

Linear Optimisation: A Mathematical Success Story...



George Bernard Dantzig
(1914—2005)



developed simplex
algorithm to solve
linear optimisation models(1947)



Leonid G. Khachiyan
(1952—2005)

Showed LP's can be solved
efficiently(1979)



Narendra K. Karmarkar
(1957—)

First practical algorithm utilizing
Khachiyan's results(1985)

Linear Optimisation: A Mathematical Success Story...

Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

Faster Solutions Seen

These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances, or whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use them in creating portfolios with the best mix of stocks and bonds.

The Bell Labs mathematician, Dr. Narendra Karmarkar, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

"This is a path-breaking result," said Dr. Ronald L. Graham, director of mathematical sciences for Bell Labs in Murray Hill, N.J.

"Science has its moments of great progress, and this may well be one of them."

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

A procedure devised in 1947, the simplex method, is now used for such problems,

Continued on Page A19, Column 1



Karmarkar at Bell Labs: an equation to find a new way through the maze

Folding the Perfect Corner

A young Bell scientist makes a major math breakthrough

Every day 1,200 American Airlines jets crisscross the U.S., Mexico, Canada and the Caribbean, stopping in 110 cities and bearing over 80,000 passengers. More than 4,000 pilots, copilots, flight personnel, maintenance workers and baggage carriers are shuffled among the flights; a total of 3.6 million gal. of high-octane fuel is burned. Nuts, bolts, altimeters, landing gears and the like must be checked at each destination. And while performing these scheduling gymnastics, the company must keep a close eye on costs, projected revenue and profits.

Like American Airlines, thousands of companies must routinely untangle the myriad variables that complicate the efficient distribution of their resources. Solving such monstrous problems requires the use of an abstruse branch of mathematics known as linear programming. It is the kind of math that has frustrated theoreticians for years, and even the fastest and most powerful computers have had great difficulty juggling the bits and pieces of data. Now Narendra Karmarkar, a 28-year-old

Indian-born mathematician at Bell Laboratories in Murray Hill, N.J., after only a year's work has cracked the puzzle of linear programming by devising a new algorithm, a step-by-step mathematical formula. He has translated the procedure into a program that should allow computers to track a greater combination of tasks than ever before and in a fraction of the time.

Unlike most advances in theoretical mathematics, Karmarkar's work will have an immediate and major impact on the real world. "Breakthrough is one of the most abused words in science," says Ronald Graham, director of mathematical sciences at Bell Labs. "But this is one situation where it is truly appropriate."

Before the Karmarkar method, linear equations could be solved only in a cumbersome fashion, ironically known as the simplex method, devised by Mathematician George Dantzig in 1947. Problems are conceived of as giant geodesic domes with thousands of sides. Each corner of a facet on the dome

...With An Unprecedented Impact On Practical Decision-Making!

Karmarkar Algorithm Proves Its Worth

Less than two years after discovery of a mathematical procedure that Bell Labs said could solve a broad range of complex business problems 50 to 100 times faster than current methods, AT&T is filing for patents covering its use. The Karmarkar algorithm, which drew headlines when discovered by researcher Narendra Karmarkar, will be applied first to AT&T's long-distance network.

Thus far, Bell Labs has verified the procedure's capabilities in developing plans for new fiber-optic transmission and satellite capacity linking 20 countries bordering the Pacific Ocean. That jointly owned network will be built during the next 10 years. Planning requires a tremendous number of "what if" scenarios involving 43,000 variables describing transmission capacity, location and construction schedules, all juggled amid political considerations of each connected country.

The Karmarkar algorithm was able to solve the Pacific Basin problem in four minutes, against 80 minutes by the method previously used, says Neil Dinn, head of Bell Labs' international transmission planning department. The speedier solutions will enable international committees to agree on network designs at one meeting instead of many meetings stretched out over months.

AT&T now is using the Karmarkar procedure to plan construction for its domestic network, a problem involving 800,000 variables. In addition, the procedure may be written into software controlling routing of domestic phone calls, boosting the capacity of AT&T's current network.

The Wall Street Journal, Jul 18, 1986

AT&T Markets Problem Solver, Based On Math Whiz's Find, for \$8.9 Million

By ROGER LOWENSTEIN

Staff Reporter of THE WALL STREET JOURNAL

NEW YORK—American Telephone & Telegraph Co. has called its math whiz, Narendra Karmarkar, a latter-day Isaac

only very large companies—mostly in the Fortune 100.

Korbx "won't have a significant bottom-line impact initially" for AT&T, though it might in the long term, says Charles Nichols, an

The Wall Street Journal, Aug 15, 1988

THE STARTLING DISCOVERY BELL LABS KEPT IN THE SHADOWS

Now its breakthrough mathematical formula could save business millions

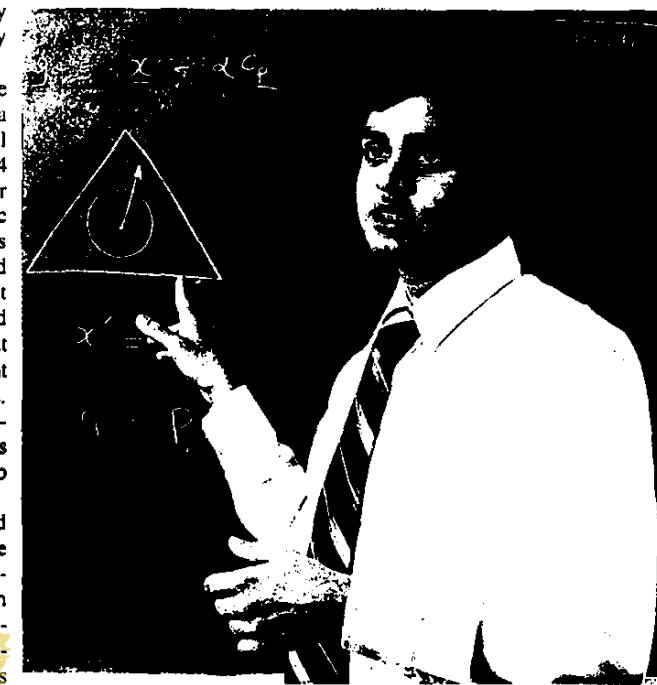
It happens all too often in science. An obscure researcher announces a stunning breakthrough and achieves instant fame. But when other scientists try to repeat his results, they fail. Fame quickly turns to notoriety, and eventually the episode is all but forgotten.

That seemed to be the case with Narendra K. Karmarkar, a young scientist at AT&T Bell Laboratories. In late 1984 the 28-year-old researcher astounded not only the scientific community but also the business world. He claimed he had cracked one of the thorniest aspects of computer-aided problem-solving. If so, his feat would have meant an instant windfall for many big companies. It could also have pointed to better software for small companies that use computers to help manage their business.

Karmarkar said he had discovered a quick way to solve problems so hideously complicated that they often defy even the most powerful supercomputers. Such problems bedevil a broad range of business activities, from assessing risk factors in stock portfolios to drawing up production schedules in factories. Just about any company that distributes products through more than a handful of warehouses bumps into such problems when calculating the cheapest routes for getting goods to customers. Even when the problems aren't terribly complex, solving them can chew up so much computer time that the answer is useless before it's found.

HEAD START. To most mathematicians, Karmarkar's precocious feat was hard to swallow. Because such questions are so common, a special branch of mathematics called

twist. Other scientists weren't able to duplicate Karmarkar's work, it turns out, because his employer wanted it that way. Vital details about how best to translate the



KARMAKAR: SKEPTICS ATTACKED HIS PRECOCIOUS FEAT

algorithm, whose mathematical notations run on for about 20 printed pages, into digital computer code were withheld to give Bell Labs a head start at developing commercial products. Following the breakup of American Telephone & Telegraph Co. in January, 1984, Bell Labs was no longer prevented from exploiting its research for profit. While the underlying concept could not be patented or copyrighted because it is pure knowledge, any computer programs that AT&T developed to implement the procedure can be protected.

Now, AT&T may soon be selling the first product based on Karmarkar's work—to the U.S. Air Force. It includes a multiprocessor computer from Alliant Computer Systems Corp. and a software version of Karmarkar's algorithm that has been optimized for high-speed parallel processing. The system would be installed at St. Louis' Scott Air

Force Base, headquarters of the Military Airlift Command (MAC). Neither party will comment on the deal's cost or where the negotiations stand, but the Air Force's interest is easy to fathom.

JUGGLING ACT. On a typical day thousands of planes ferry cargo and passengers among air fields scattered around the world. To keep those jets flying, MAC

linear programming (LP) has evolved, and most scientists thought that was as far as they could go. Sure enough, when other researchers independently tried to test Karmarkar's process, their results were disappointing. At scientific conferences skeptics attacked the algorithm's validity as well as Karmarkar's veracity.

Business Week, Sep 21, 1987

Production Planning

The Winnipeg plant of the *Gemstone Tool Company* (GTC) produces wrenches and pliers. Both tools are made from steel, and the production requires (i) molding the tools on a molding machine and (ii) assembling the tools on an assembly machine. GTC wants to determine the daily production of wrenches and pliers so as to maximise their contribution to earnings, subject to the daily demand limits.

	Wrenches	Pliers	Availability
Required steel (lbs.)	1,5	1,0	27,000 lbs./day
Required time on molding machine (h)	1,0	1,0	21,000 h/day
Required time on assembly machine (h)	0,3	0,5	9,000 h/day
Demand limit (units/day)	15.000	16.000	
Contribution to earnings (\$/unit)	US\$ 0,13	US\$ 0,10	



- 1 What is GTC's optimal daily production plan?
- 2 What are the daily contributions to earnings from this plan?
- 3 Which resources would be most critical in this plan?

Production Planning

Step 1: Identify the *decision variables*

What decision(s) must be made in order to solve the problem?

Production Planning

Step 1: Identify the *decision variables*

What decision(s) must be made in order to solve the problem?

W = number of *wrenches* produced per day
 P = number of *pliers* produced per day

Production Planning

Step 2: Identify the *objective function*

Typically either “maximise revenue - cost”
or “minimise cost - revenue”

Step 2: Identify the *objective function*

Typically either “maximise revenue - cost”
or “minimise cost - revenue”

Maximise the contribution to earnings

maximise $0.13W + 0.10P$

Production Planning

Step 3: Identify the *constraints*

What hinders me from making infinite revenues or incurring zero costs?

Step 3: Identify the *constraints*

What hinders me from making infinite revenues or incurring zero costs?

$$1.5W + 1.0P \leq 27,000$$

“Availability of steel”

$$1.0W + 1.0P \leq 21,000$$

“Availability of molding machine”

$$0.3W + 0.5P \leq 9,000$$

“Availability of assembly machine”

$$W \leq 15,000, \quad P \leq 16,000$$

“Daily demand limits”

$$W, P \geq 0$$

“We cannot produce negative amounts of tools”

The overall linear optimisation problem is:

maximise $0.13W + 0.10P$

subject to $1.5W + 1.0P \leq 27,000$

$1.0W + 1.0P \leq 21,000$

$0.3W + 0.5P \leq 9,000$

$W \leq 15,000, P \leq 16,000$

$W, P \geq 0$

The optimal solution to this problem is:

Product	Decision Variable	Production amount
Wrenches	W	12.000
Pliers	P	9.000



The overall daily contribution to earnings is:

$$\$0.13W + \$0.10P = \$2,460$$

The binding resources are:

steel: $1.5W + 1.0P = 27,000$

molding machine: $1.0W + 1.0P = 21,000$

assembly machine: $0.3W + 0.5P = 8,100 < 9,000$



Solving Linear Programs: Graphical Solution

Recall GTC's linear program:

maximise $0.13W + 0.10P$

subject to $1.5W + 1.0P \leq 27,000$

$1.0W + 1.0P \leq 21,000$

$0.3W + 0.5P \leq 9,000$

$W \leq 15,000, P \leq 16,000$

$W, P \geq 0$

Solving Linear Programs: Graphical Solution

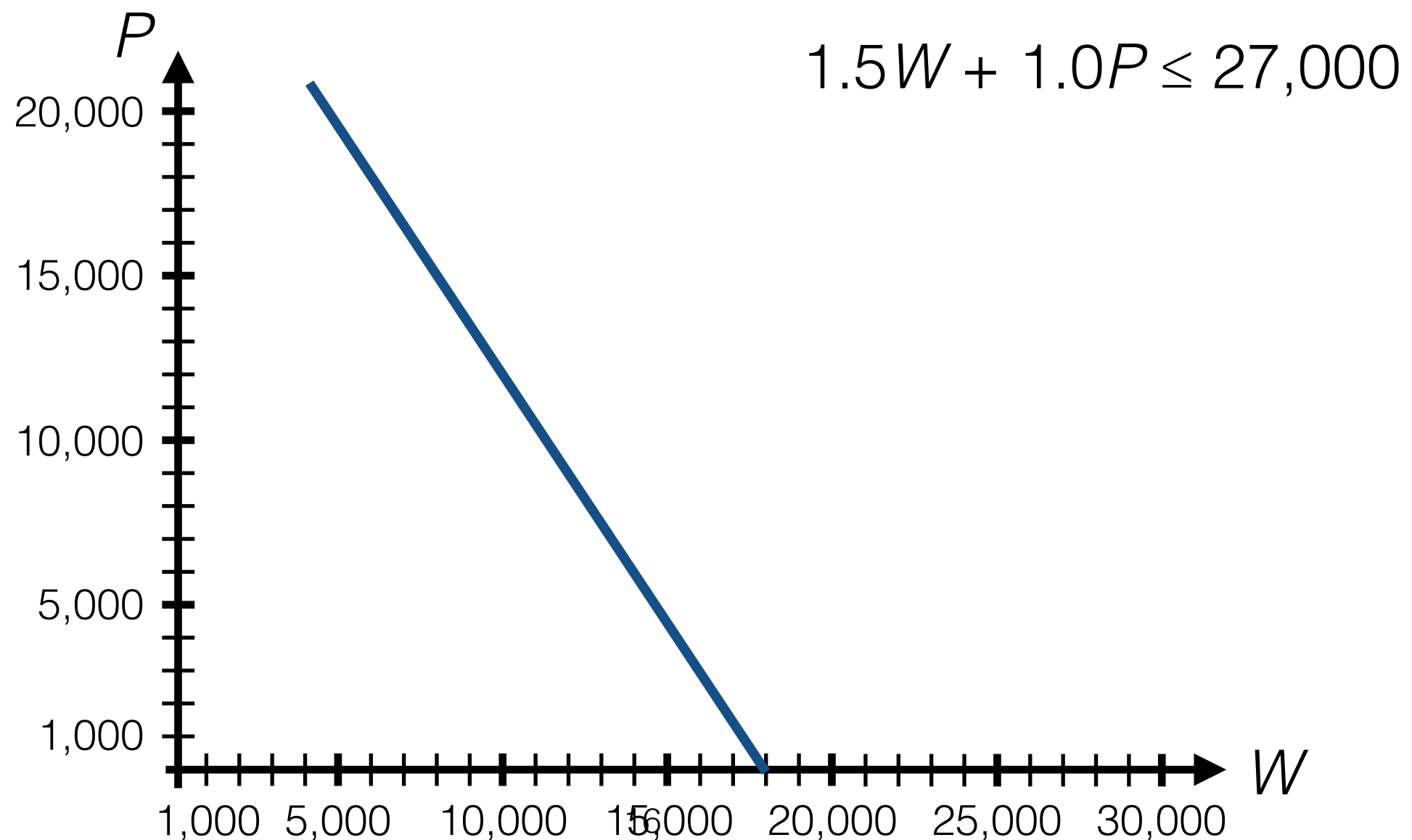
Step 1: Plot each of the constraints

- plot the constraint equation line
- determine which of the two regions satisfies the constraint

Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

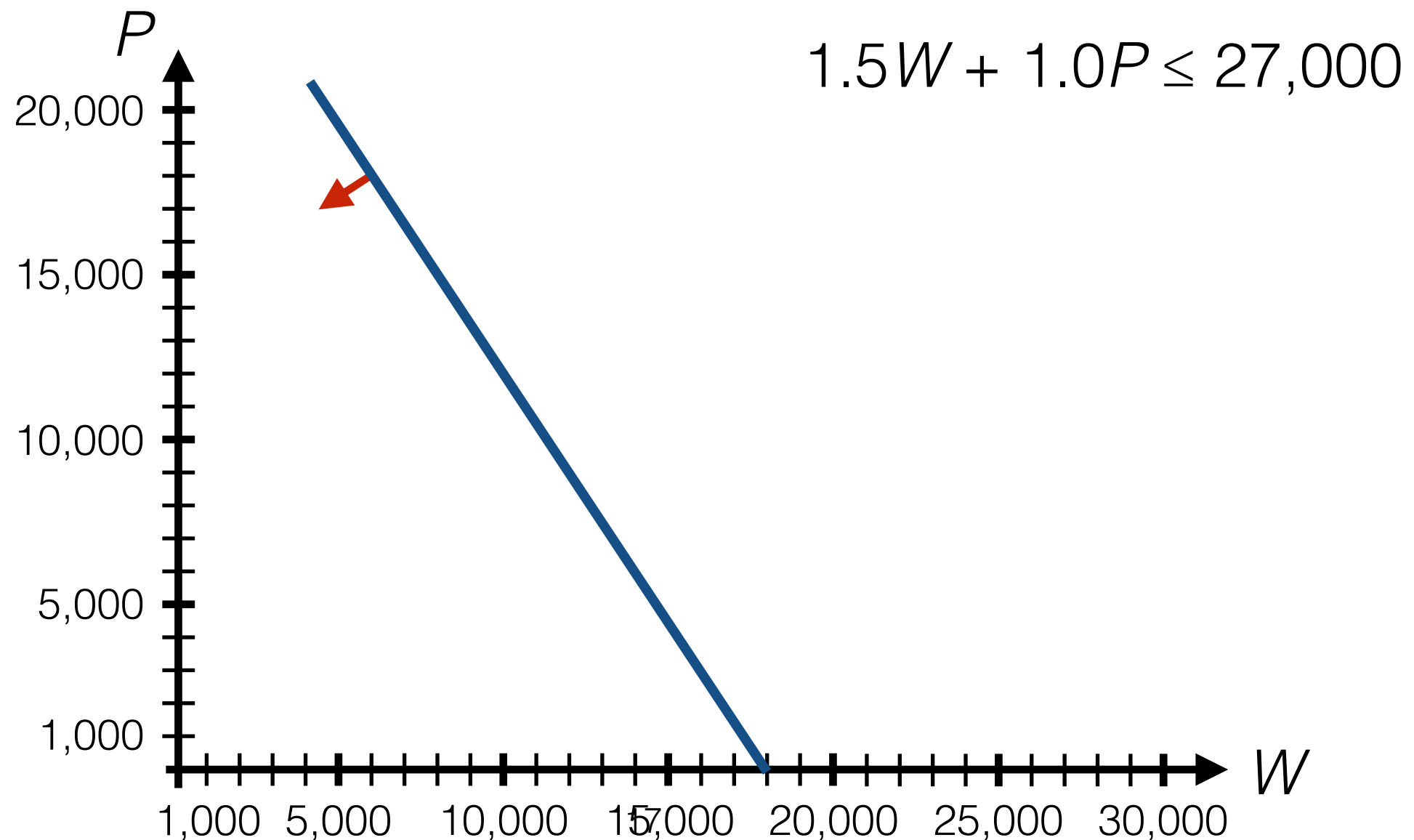
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

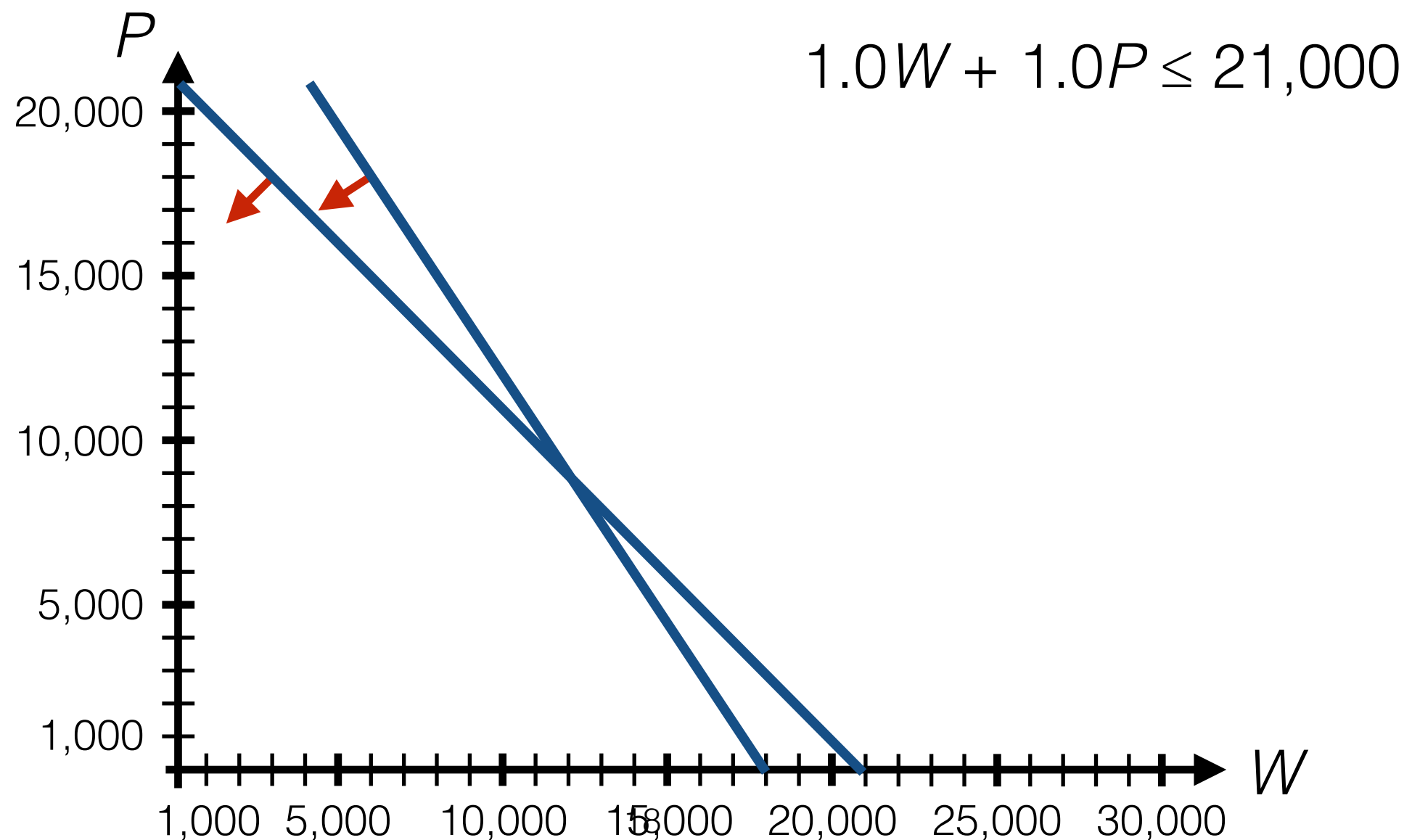
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

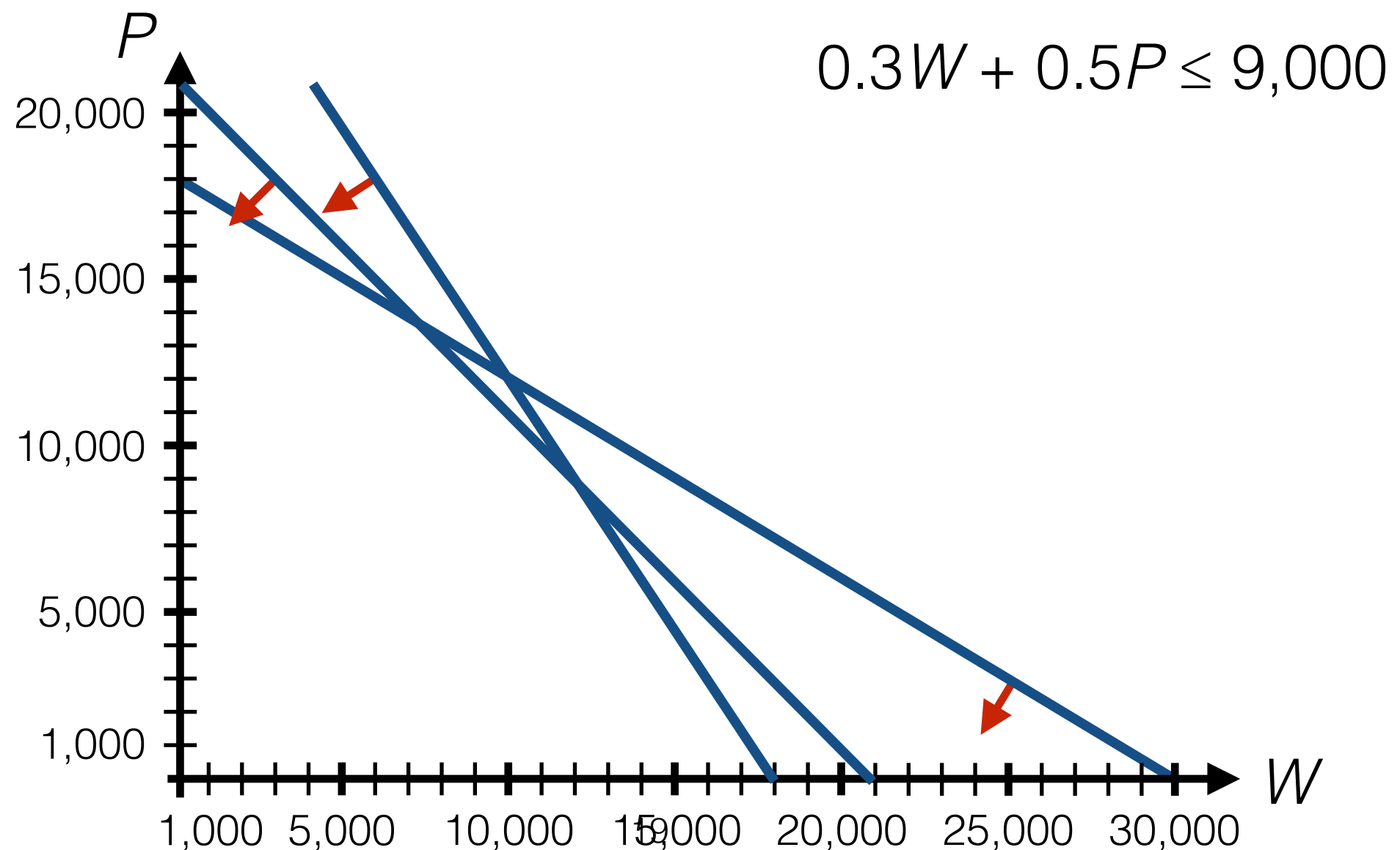
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

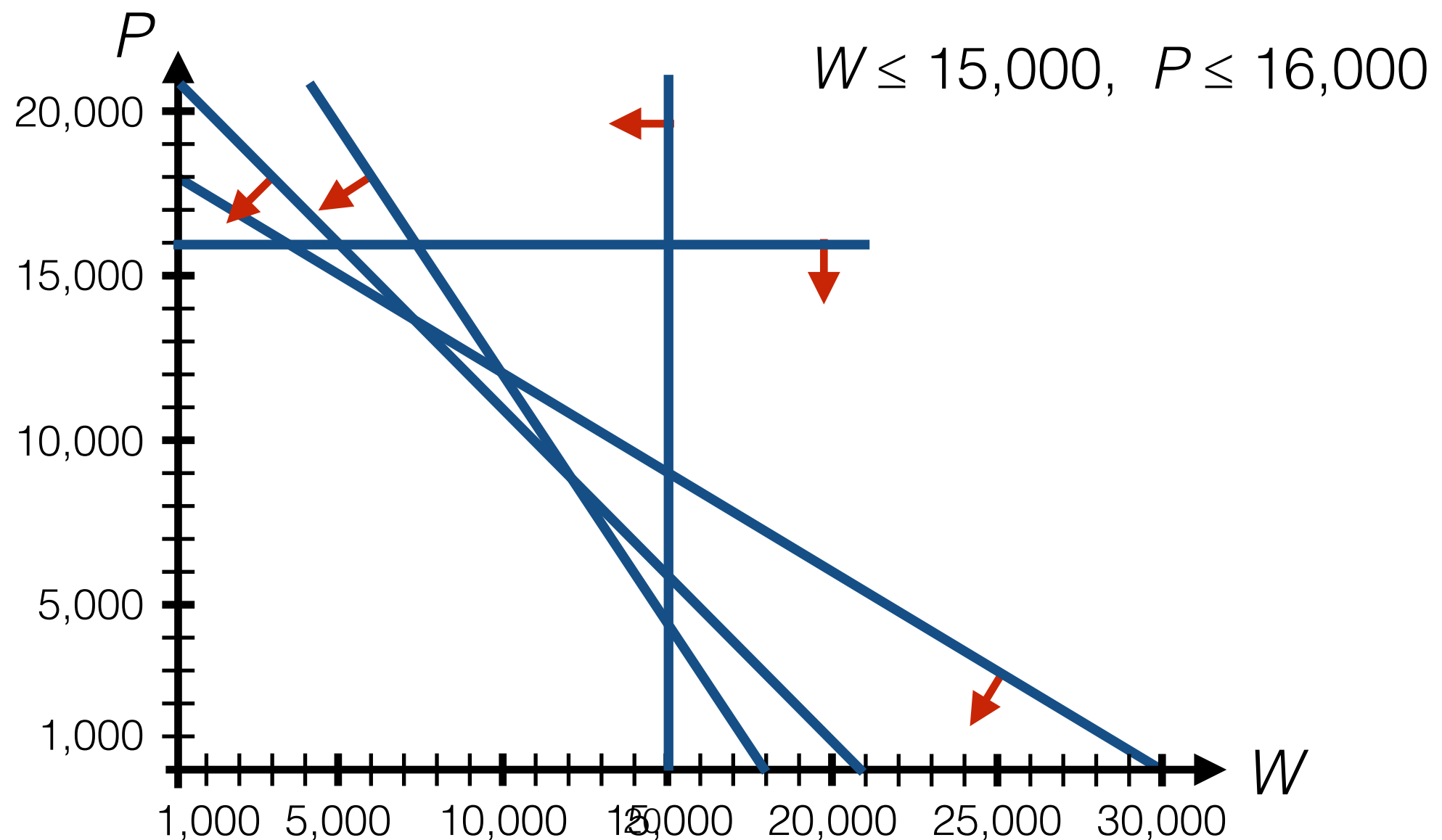
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

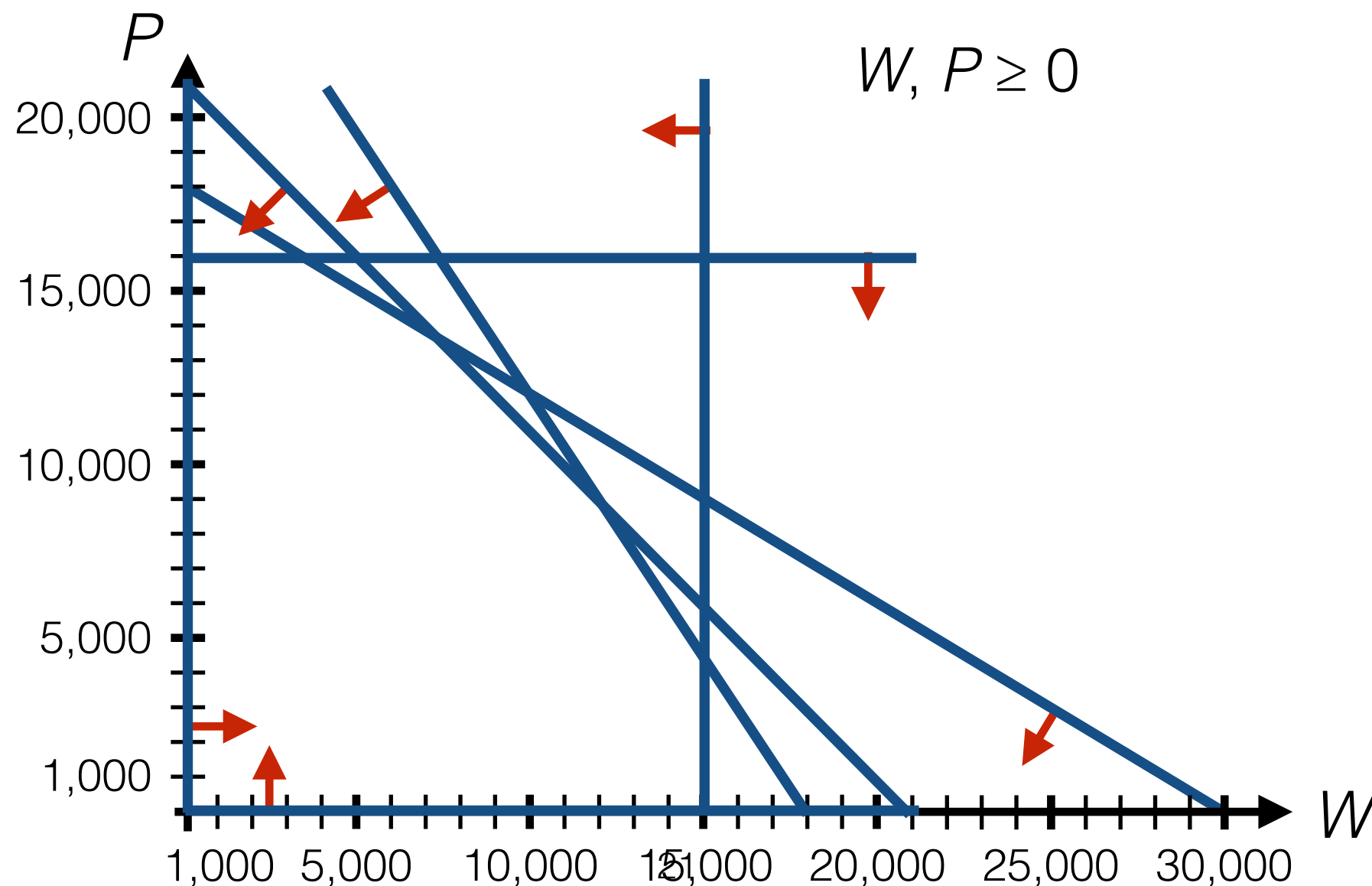
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 1: Plot each of the constraints

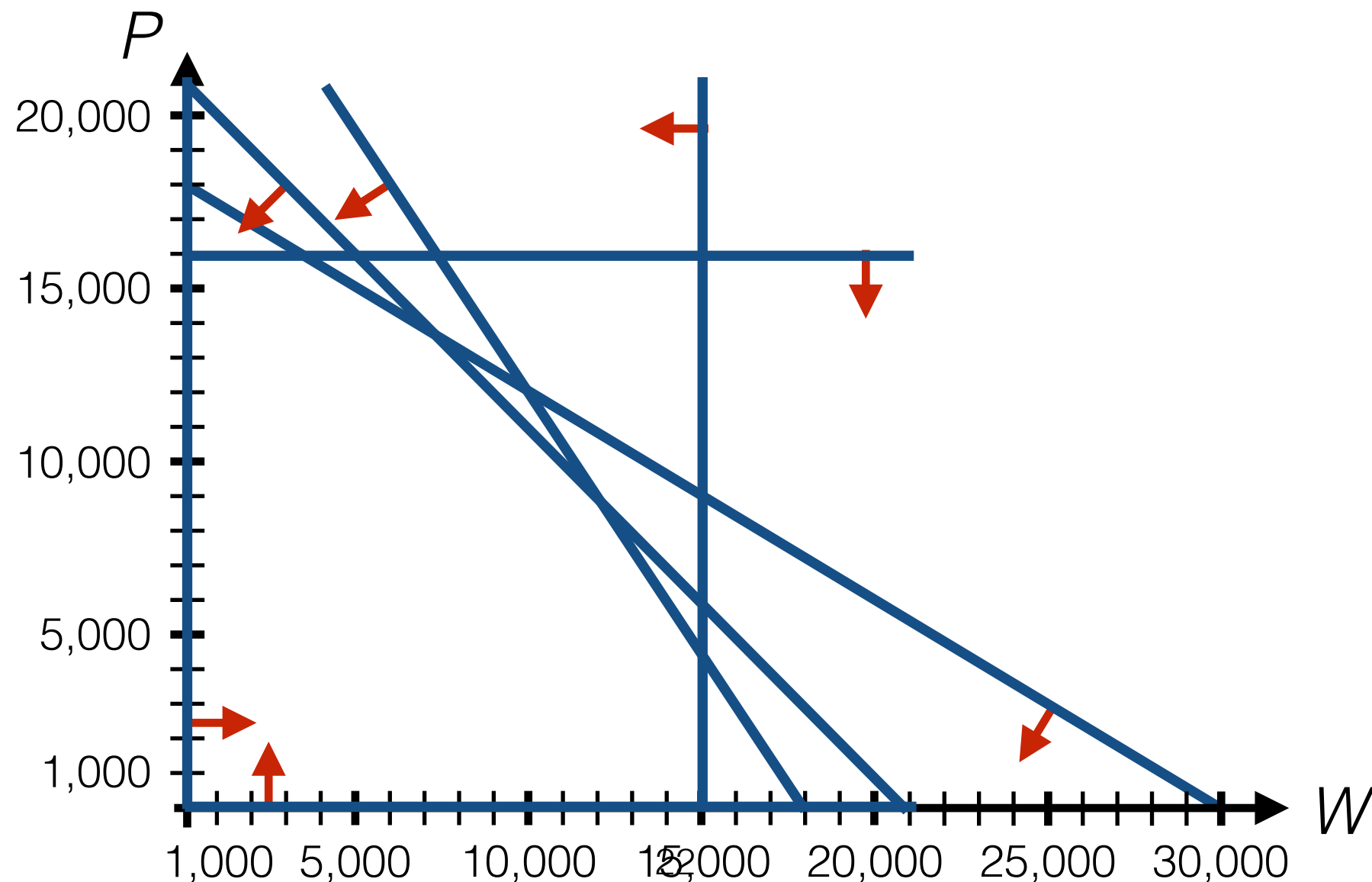
- plot the constraint equation line
- determine which of the two regions satisfies the constraint



Solving Linear Programs: Graphical Solution

Step 2: Plot the feasible region

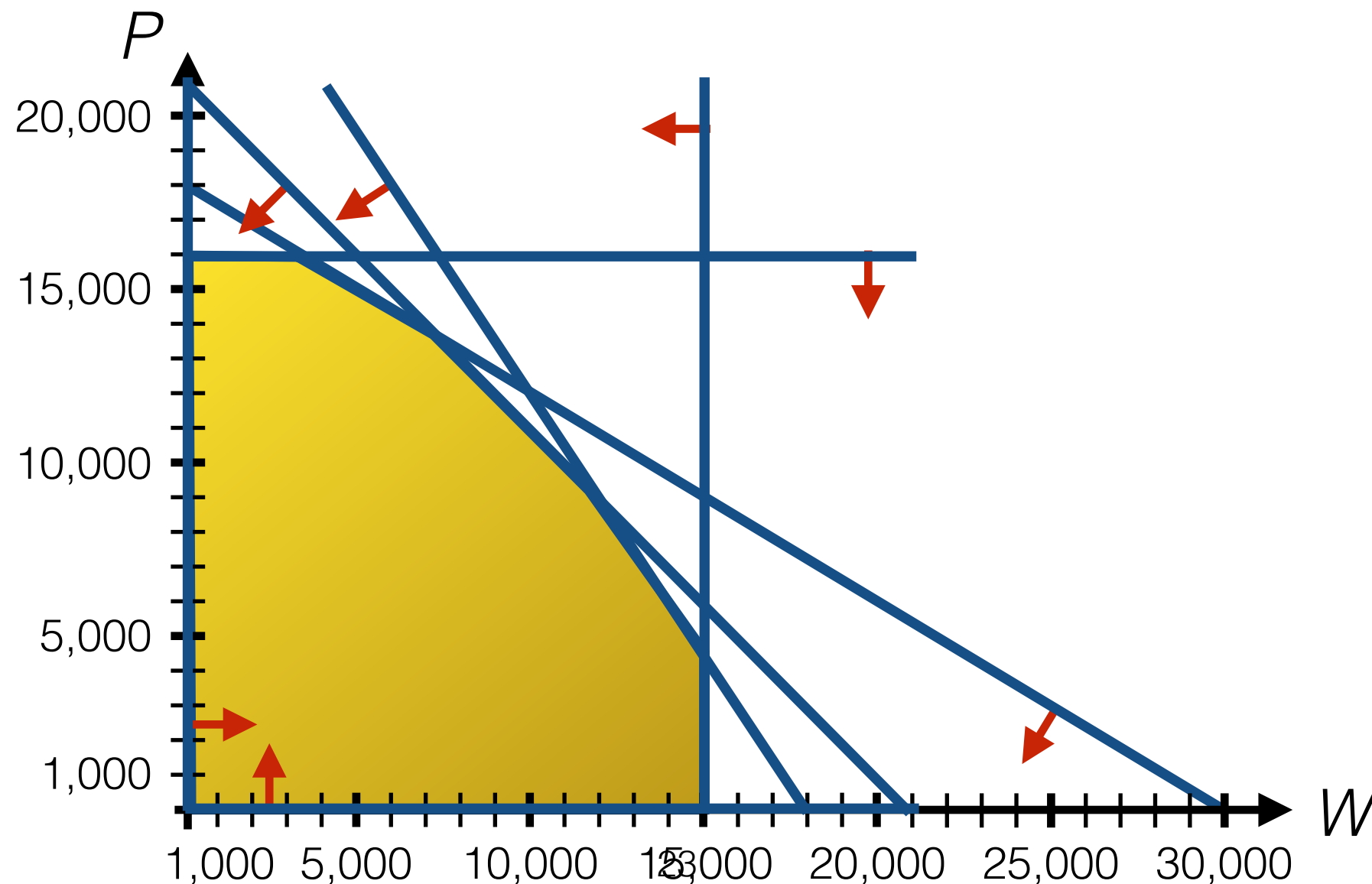
Check which area in the plane satisfies all constraints.



Solving Linear Programs: Graphical Solution

Step 2: Plot the feasible region

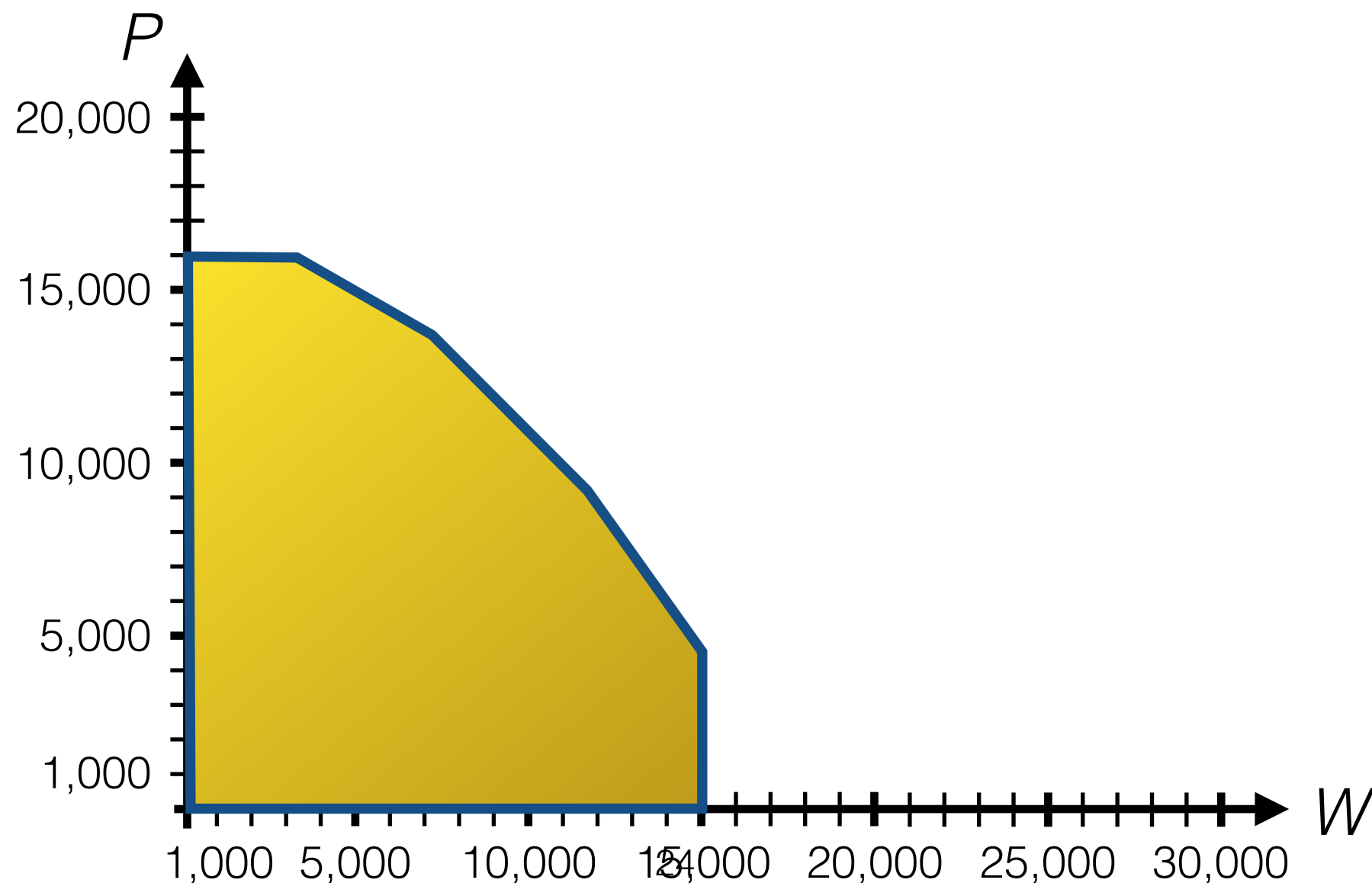
Check which area in the plane satisfies all constraints.



Solving Linear Programs: Graphical Solution

Step 3: Determine objective function isoquants

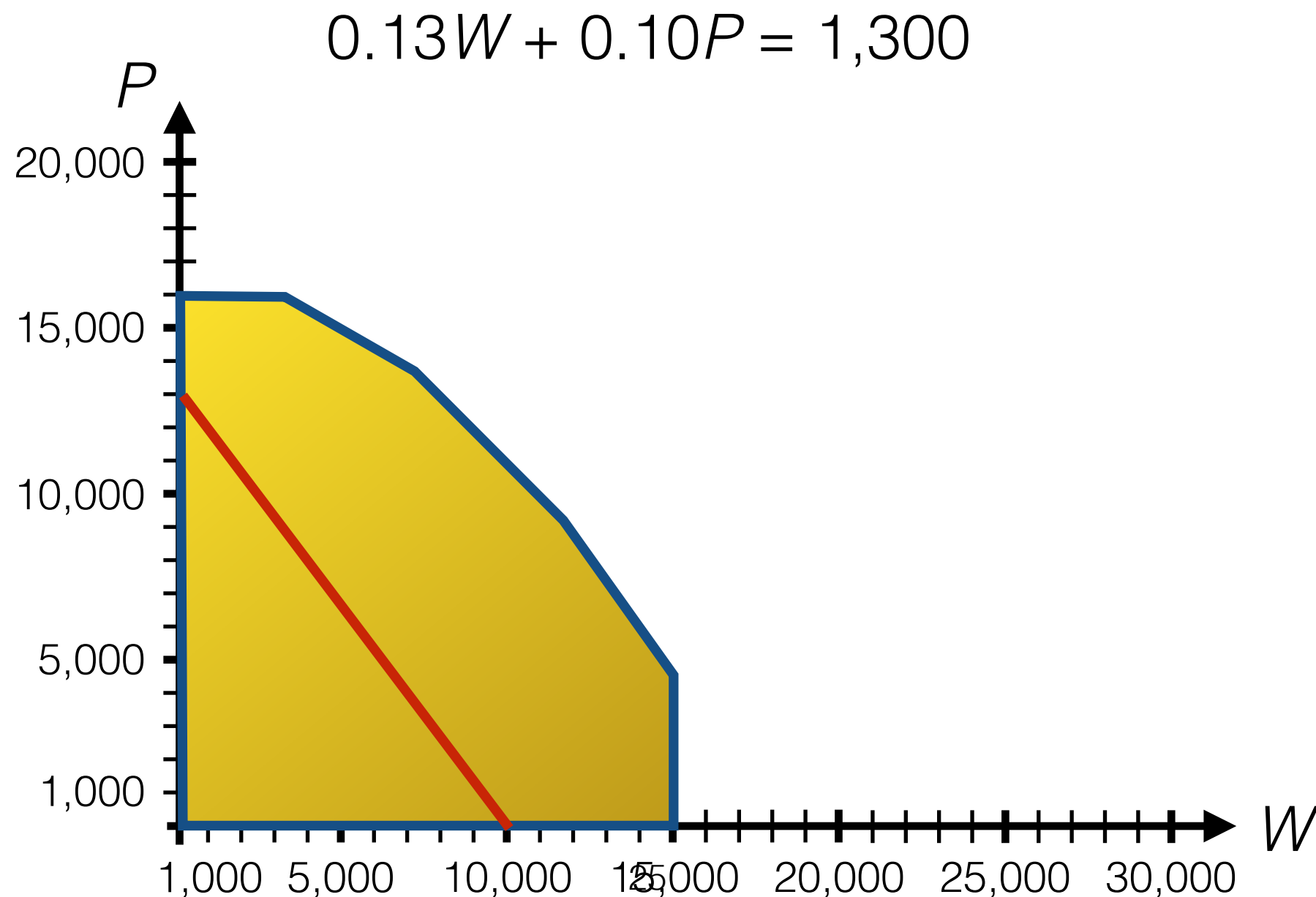
Draw all solutions that achieve a particular, fixed objective value.



Solving Linear Programs: Graphical Solution

Step 3: Determine objective function isoquants

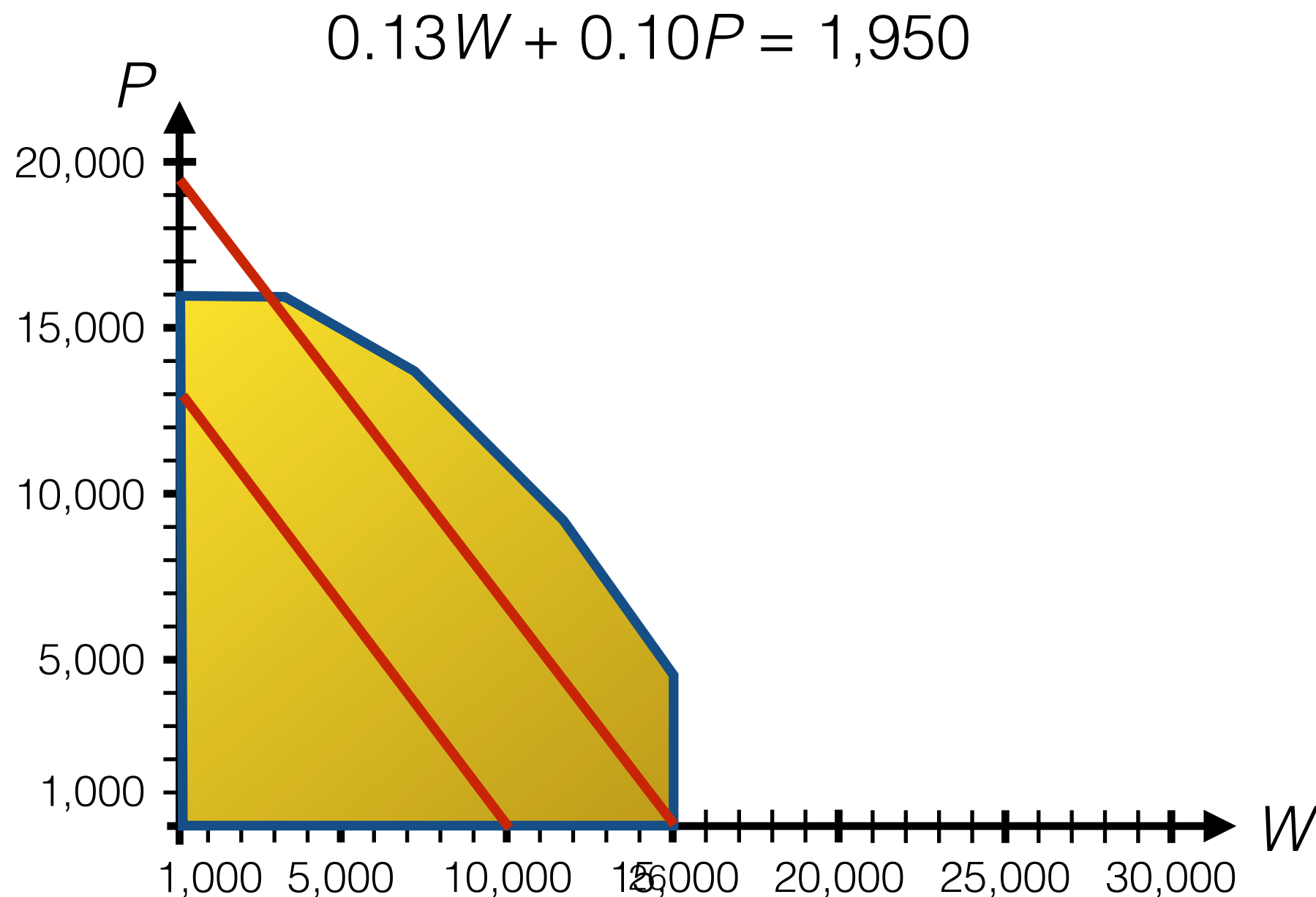
Draw all solutions that achieve a particular, fixed objective value.



Solving Linear Programs: Graphical Solution

Step 3: Determine objective function isoquants

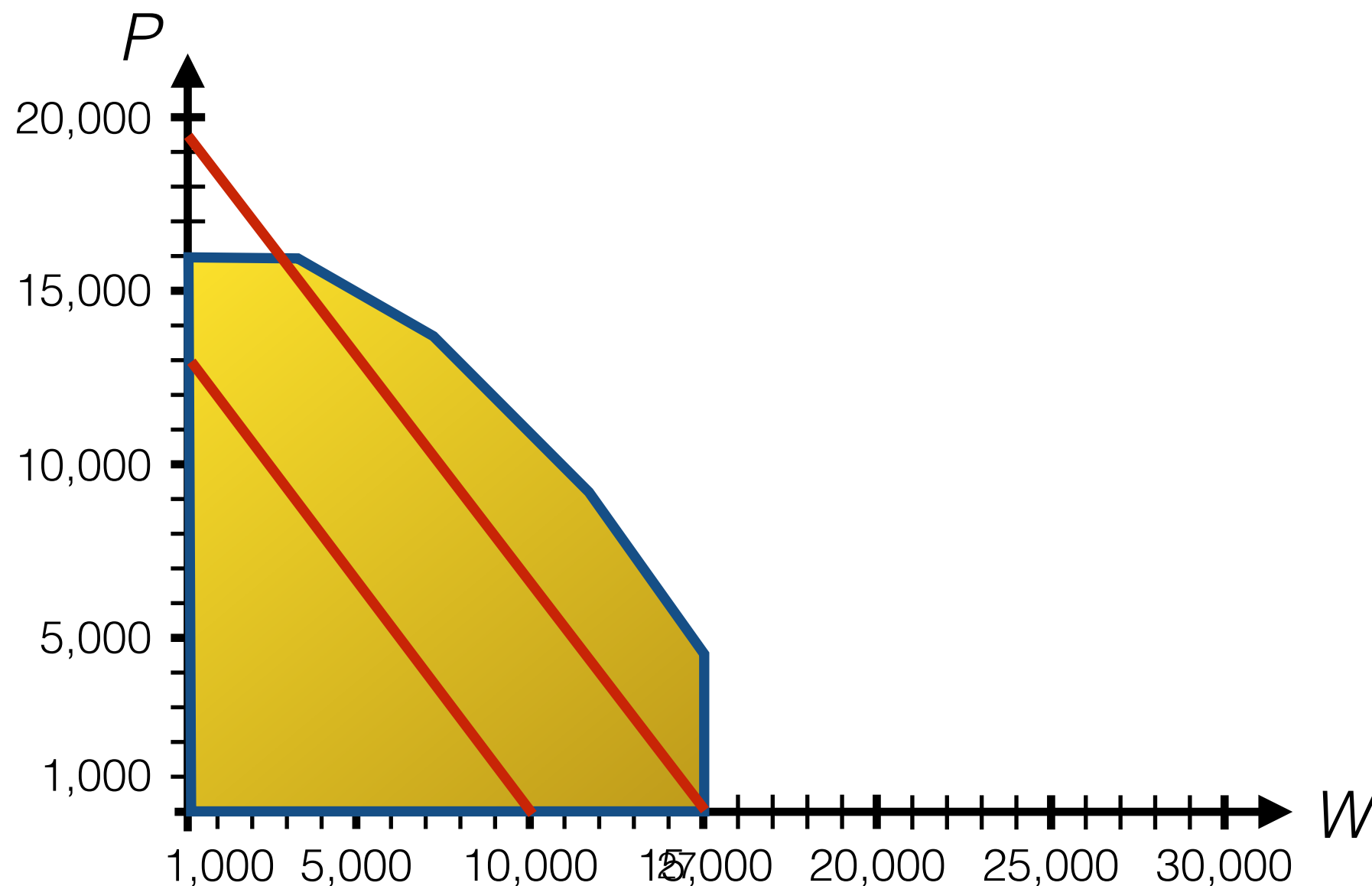
Draw all solutions that achieve a particular, fixed objective value.



Solving Linear Programs: Graphical Solution

Step 4: Determine optimal solution

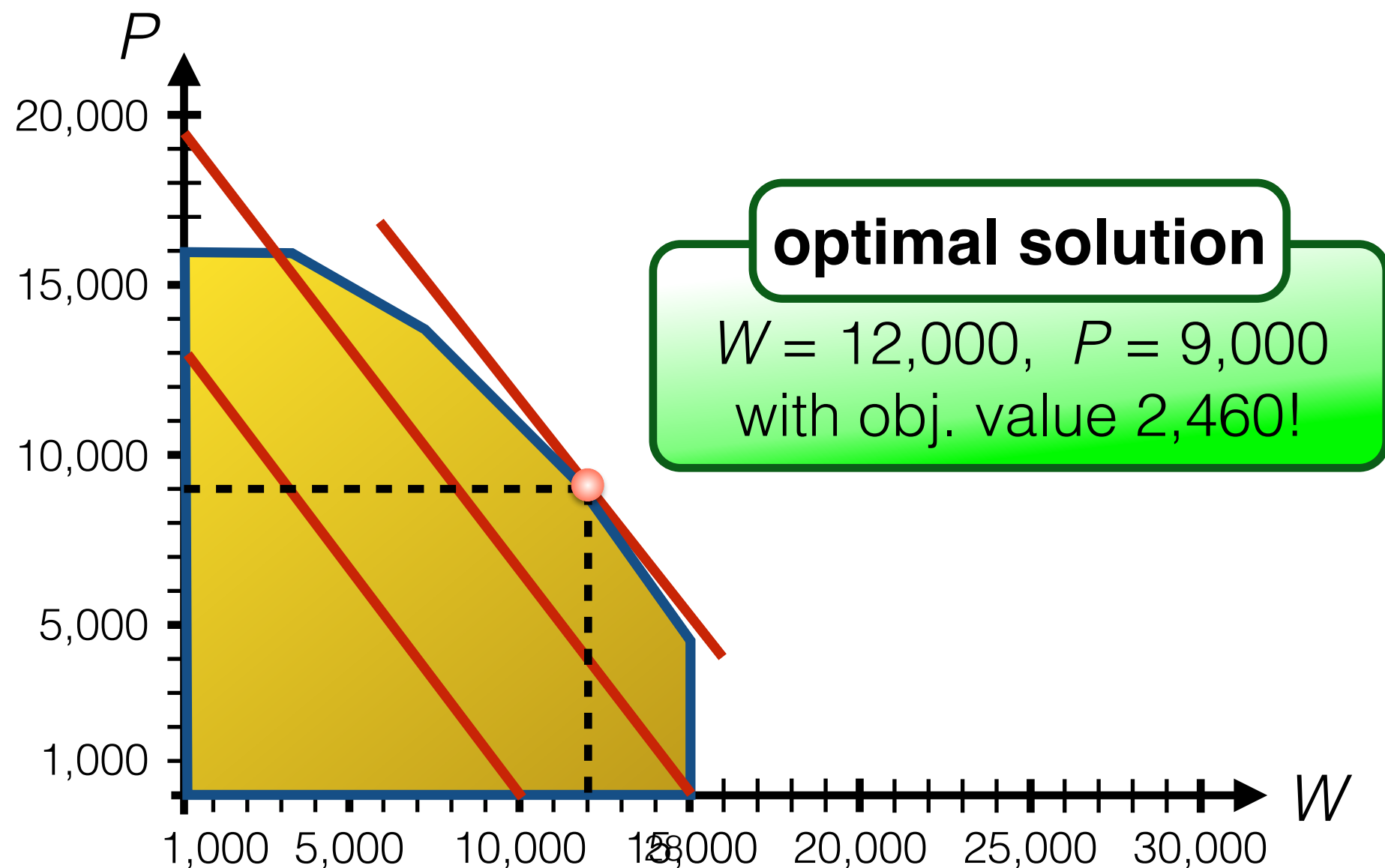
Move the isoquants in the direction of improving objective values until they just touch the feasible region.



Solving Linear Programs: Graphical Solution

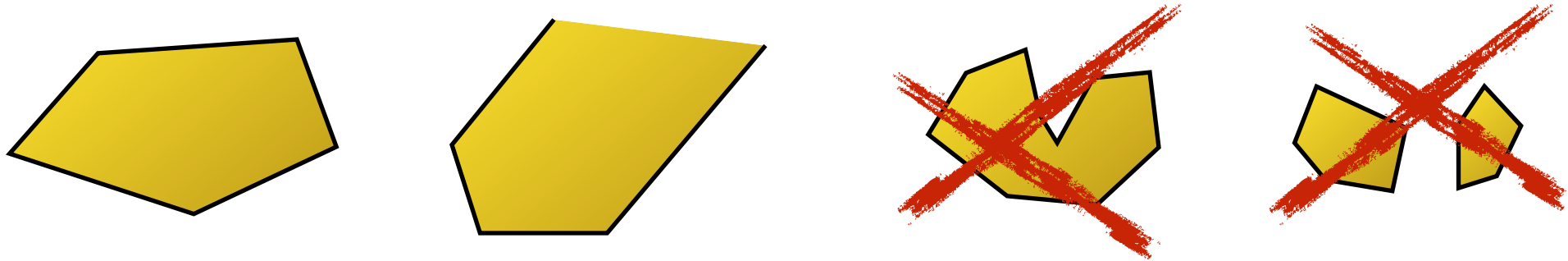
Step 4: Determine optimal solution

Move the isoquants in the direction of improving objective values until they just touch the feasible region.

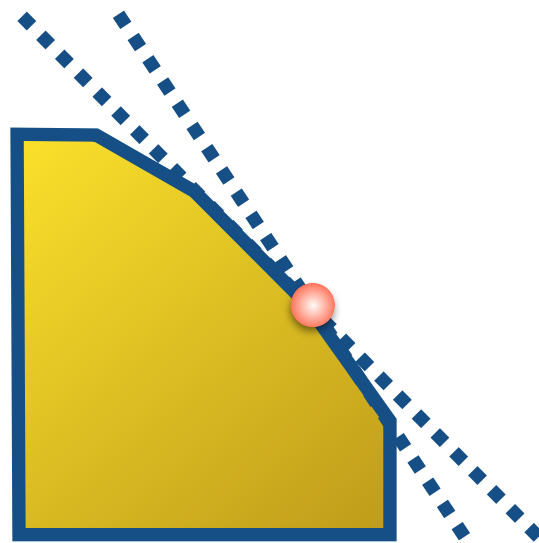


Solving Linear Programs: Insights From Graphical Solution

- 1 The feasible region of a linear program forms a *polygon*.



- 2 The optimal solution solves a *system of linear equations*.



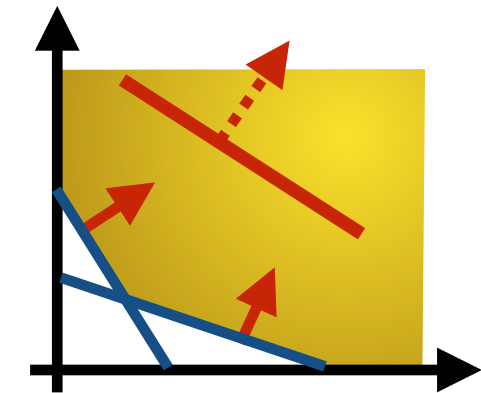
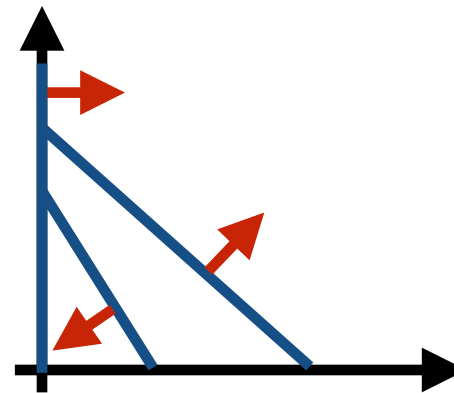
$$\begin{aligned} 1.5W + 1.0P &\stackrel{!}{=} 27,000 \\ 1.0W + 1.0P &\stackrel{!}{=} 21,000 \end{aligned}$$

- 3 At the optimal solution, there can be
- *binding constraints*: satisfied as equalities
 - *non-binding constraints*: satisfied as strict inequalities

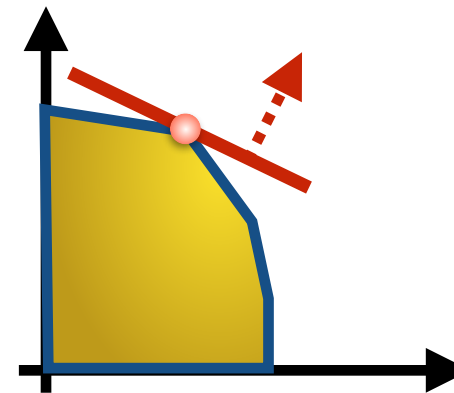
Solving Linear Programs: Insights From Graphical Solution

4 A linear program can have

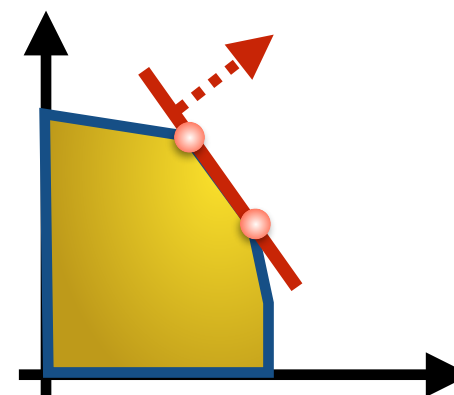
- *no optimal solution:*



- *exactly one optimal solution:*



- *infinitely many optimal solutions:*

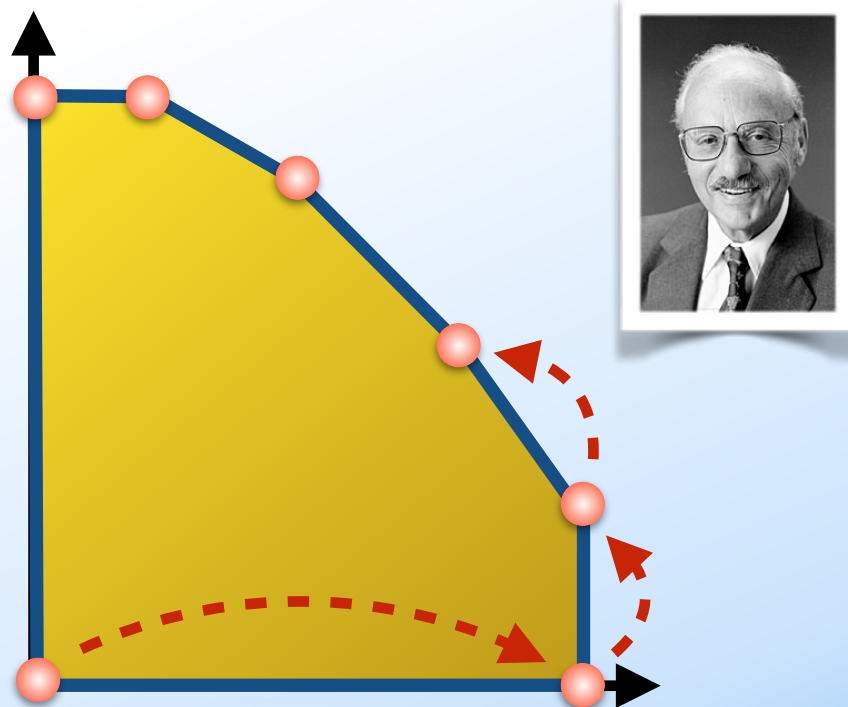


whenever there is
an optimal solution,
there is at least one
optimal *vertex*
solution (solving an
equation system)!

Solving Linear Programs: Simplex & Interior Point Methods

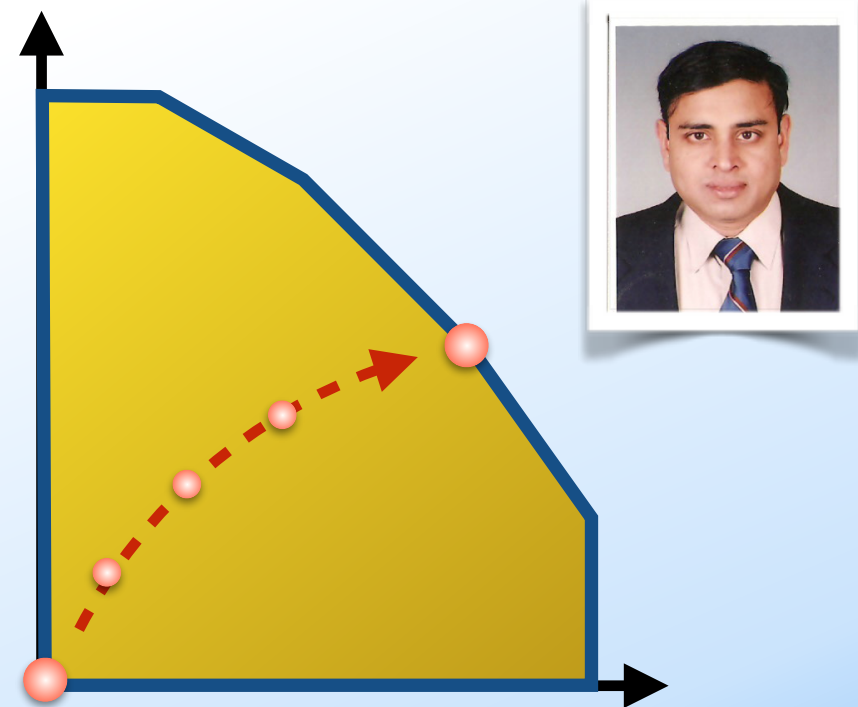
Graphical method only works for problems with ≤ 2 variables.

Simplex method



- moves from one vertex to the next, until no further improvement of objective is possible
- requires iterated solution of systems of linear equations

Interior point method



- moves through the interior of the feasible region
- comes with better solution time guarantees (an LP can have a huge number of vertices!)

Solving an LP in Gurobi

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

```
: from gurobipy import *  
model=Model()
```


Solving an LP in Gurobi

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)
```

Solving an LP in Gurobi

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)  
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)
```

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)  
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)  
model.addConstr(1.5*wrenches+1.0*plyers<=27000)  
model.addConstr(wrenches+plyers<=21000)  
model.addConstr(0.3*wrenches+0.5*plyers<=9000)
```

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)  
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)  
model.addConstr(1.5*wrenches+1.0*plyers<=27000)  
model.addConstr(wrenches+plyers<=21000)  
model.addConstr(0.3*wrenches+0.5*plyers<=9000)  
wrenches.UB=15000  
plyers.UB=16000  
wrenches.LB=0  
plyers.LB=0
```

maximise $0.13W + 0.10P$
 subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

```

: from gurobipy import *
model=Model()
wrenches=model.addVar(vtype=GRB.CONTINUOUS)
plyers=model.addVar(vtype=GRB.CONTINUOUS)
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)
model.addConstr(1.5*wrenches+1.0*plyers<=27000)
model.addConstr(wrenches+plyers<=21000)
model.addConstr(0.3*wrenches+0.5*plyers<=9000)
wrenches.UB=15000
plyers.UB=16000
wrenches.LB=0
plyers.LB=0
model.optimize()

```

Academic license - for non-commercial use only

Optimize a model with 3 rows, 2 columns and 6 nonzeros

Coefficient statistics:

Matrix range [3e-01, 2e+00]
 Objective range [1e-01, 1e-01]
 Bounds range [2e+04, 2e+04]
 RHS range [9e+03, 3e+04]

Presolve time: 0.01s

Presolved: 3 rows, 2 columns, 6 nonzeros

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	2.7000000e+03	3.250000e+03	0.000000e+00	0s
3	2.4600000e+03	0.000000e+00	0.000000e+00	0s

Solved in 3 iterations and 0.02 seconds

Optimal objective 2.460000000e+03

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)  
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)  
model.addConstr(1.5*wrenches+1.0*plyers<=27000)  
model.addConstr(wrenches+plyers<=21000)  
model.addConstr(0.3*wrenches+0.5*plyers<=9000)  
wrenches.UB=15000  
plyers.UB=16000  
wrenches.LB=0  
plyers.LB=0  
model.optimize()
```

Academic license - for non-commercial use only
Optimize a model with 3 rows, 2 columns and 6 nonzeros

```
print ("wrenches to produce",wrenches.X)  
print ("plyers to produce",plyers.X)  
print("optimal revenue",model.objVal)
```

```
wrenches to produce 12000.0  
plyers to produce 8999.999999999998  
optimal revenue 2460.0
```

Solving an LP in Gurobi

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Variable attributes:

These attributes provide information that is associated with specific variables.

Attribute name	Description
LB	Lower bound
UB	Upper bound
Obj	Linear objective coefficient
VType	Variable type (continuous, binary, integer, etc.)
VarName	Variable name
VTag	Variable tag

These attributes provide information that is associated with specific linear constraints.

Attribute name	Description
Sense	Constraint sense ('<', '>', or '=')
RHS	Right-hand side value
ConstrName	Constraint name
Ctag	Constraint tag
Pi	Dual value (also known as the <i>shadow price</i>)
Slack	Slack in the current solution

```
print ("wrenches to produce",wrenches.X)
print ("plyers to produce",plyers.X)
print("optimal revenue",model.objVal)
```

```
wrenches to produce 12000.0
plyers to produce 8999.999999999998
optimal revenue 2460.0
```

Look in the reference manual for attribute names!

maximise $0.13W + 0.10P$
subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

Poor style

```
: from gurobipy import *  
model=Model()  
wrenches=model.addVar(vtype=GRB.CONTINUOUS)  
plyers=model.addVar(vtype=GRB.CONTINUOUS)  
model.setObjective(0.13*wrenches+0.1*plyers,GRB.MAXIMIZE)  
model.addConstr(1.5*wrenches+1.0*plyers<=27000)  
model.addConstr(wrenches+plyers<=21000)  
model.addConstr(0.3*wrenches+0.5*plyers<=9000)  
wrenches.UB=15000  
plyers.UB=16000
```

Better style

- Separate model from data

 - Model once, resolve for different data

- Use (and translate to Gurobi) mathematical notation

 - Easier to handle large problems

maximise $0.13W + 0.10P$
 subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Solving an LP in Gurobi

Data

Resource consumption			
	wrenches	pliers	available
steel	1,5	1	27000
molding	1	1	21000
assembling	0,3	0,5	9000
demands	15000	16000	
prices	0,13	0,1	

Solving an LP in Gurobi

maximise $0.13W + 0.10P$
 subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Data

Resource consumption			
	wrenches	pliers	available
steel	1,5	1	27000
molding	1	1	21000
assembling	0,3	0,5	9000
demands	15000	16000	
prices	0,13	0,1	

Model

maximize $\sum_{j \in T} p_j x_j$
 subject to $\sum_{j \in T} r_{ij} x_j \leq a_i$, for all $i \in R$
 $x_j \leq d_j$, for all $j \in T$

maximise $0.13W + 0.10P$
 subject to $1.5W + 1.0P \leq 27,000$
 $1.0W + 1.0P \leq 21,000$
 $0.3W + 0.5P \leq 9,000$
 $W \leq 15,000, P \leq 16,000$
 $W, P \geq 0$

Data

Resource consumption

	wrenches	pliers	available
steel	1,5	1	27000
molding	1	1	21000
assembling	0,3	0,5	9000
demands	15000	16000	
prices	0,13	0,1	

Model

maximize $\sum_{j \in T} p_j x_j$
 subject to $\sum_{j \in T} r_{ij} x_j \leq a_i$ for all $i \in R$
 $x_j \leq d_j$ for all $j \in T$

Solving an LP in Gurobi

```

from gurobi import *
tools=["wrenches","pliers"]
resources=["steel","molding","assembling"]
r={("steel","wrenches"):1.5,("steel","pliers"):1,
   ("molding","wrenches"):1,("molding","pliers"):1,
   ("assembling","wrenches"):0.3,("assembling","pliers"):0.5}
availability={"steel":27000,"molding":21000,"assembling":9000}
demand={"wrenches":15000,"pliers":16000}
price={"wrenches":0.13,"pliers":0.1}

```

$r_{assembling,pliers}$

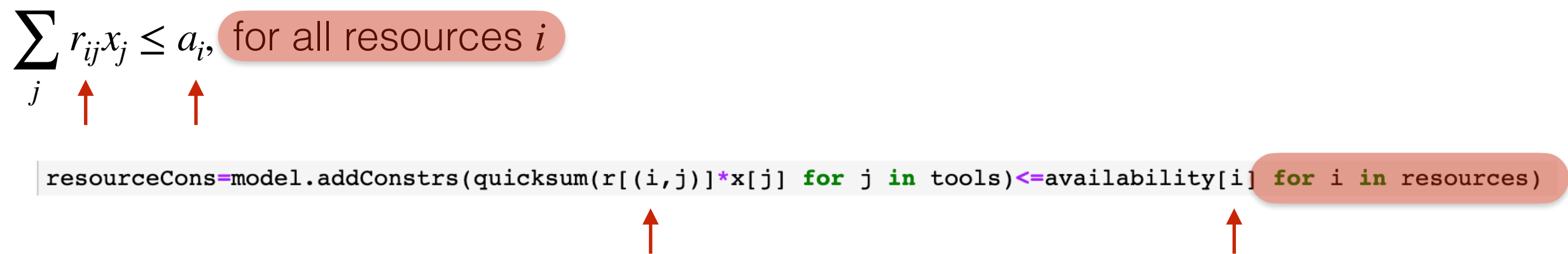
```

model=Model()
x=model.addVars(tools,obj=price,ub=demand)
model.ModelSense=GRB.MAXIMIZE
resourceCons=model.addConstrs(quicksum(r[(i,j)]*x[j] for j in tools)<=availability[i] for i in resources)
model.optimize()

```

Anatomy of the constraints

- 1 How many constraints?
-one for every resource

$$\sum_j r_{ij} x_j \leq a_i, \text{ for all resources } i$$


```
resourceCons=model.addConstrs(quicksum(r[(i,j)]*x[j] for j in tools)<=availability[i] for i in resources)
```

Anatomy of the constraints

- 1 How many constraints?
-one for every resource
- 2 Sum over which variables in each constraint?

$$\sum_j r_{ij} x_j \leq a_i, \text{ for all resources } i$$

constraint for i

```
resourceCons=model.addConstrs(quicksum(r[(i,j)]*x[j] for j in tools)<=availability[i] for i in resources)
```

Solving an LP in Gurobi

Poor style

```
: from gurobipy import *
model=Model()
wrenches=model.addVar(vtype=GRB.CONTINUOUS)
pliers=model.addVar(vtype=GRB.CONTINUOUS)
model.setObjective(0.13*wrenches+0.1*pliers,GRB.MAXIMIZE)
model.addConstr(1.5*wrenches+1.0*pliers<=27000)
model.addConstr(wrenches+pliers<=21000)
model.addConstr(0.3*wrenches+0.5*pliers<=9000)
wrenches.UB=15000
pliers.UB=16000
```

Better style

```
from gurobi import *
tools=["wrenches","pliers"]
resources=["steel","molding","assembling"]
r={("steel","wrenches"):1.5,("steel","pliers"):1,
   ("molding","wrenches"):1,("molding","pliers"):1,
   ("assembling","wrenches"):0.3,("assembling","pliers"):0.5}
availability={"steel":27000,"molding":21000,"assembling":9000}
demand={"wrenches":15000,"pliers":16000}
price={"wrenches":0.13,"pliers":0.1}
```

```
model=Model()
x=model.addVars(tools,obj=price,ub=demand)
model.ModelSense=GRB.MAXIMIZE
resourceCons=model.addConstrs(quicksum(r[(i,j)]*x[j] for j in tools)<=availability[i] for i in resources)
model.optimize()
```