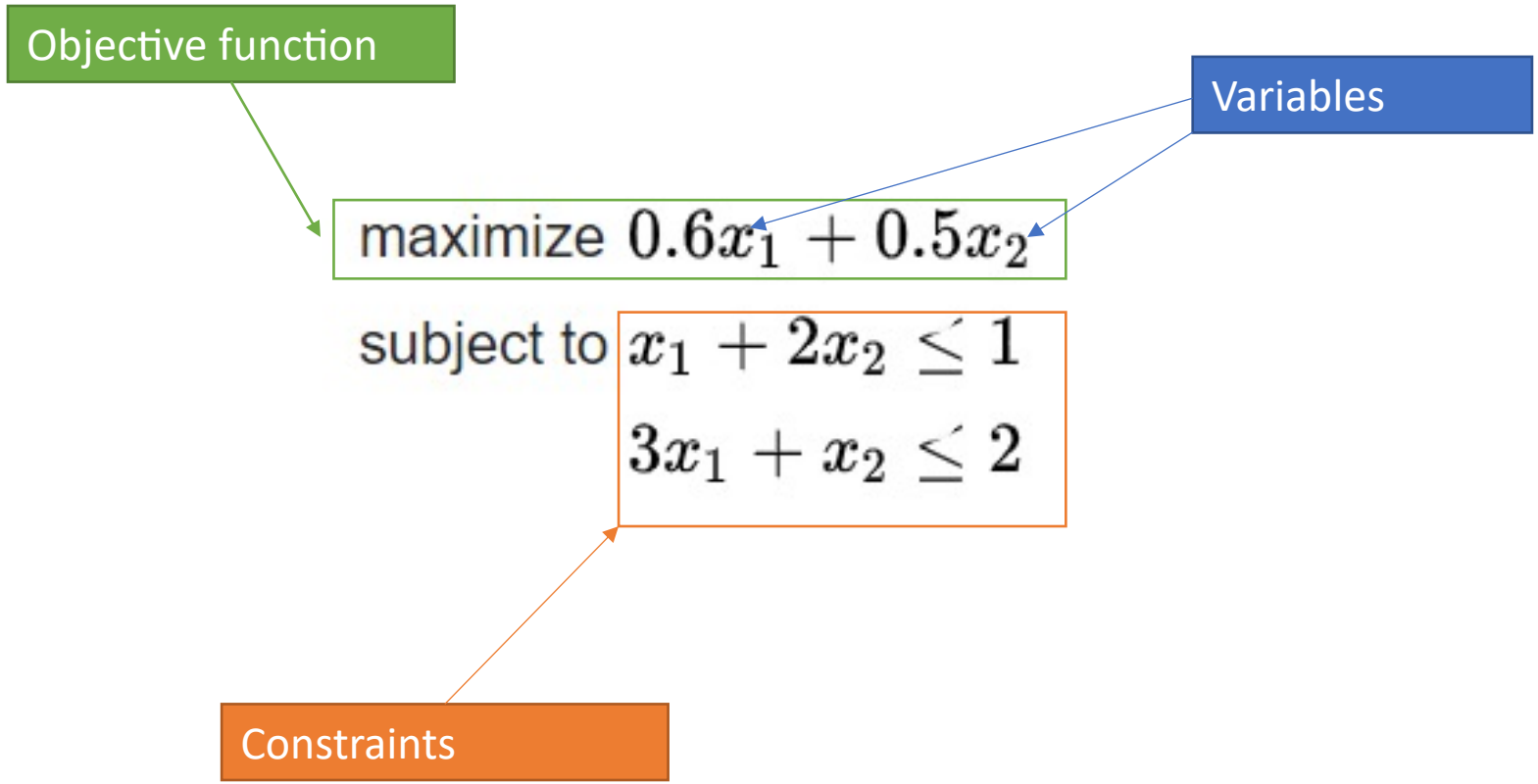
Computing Lab

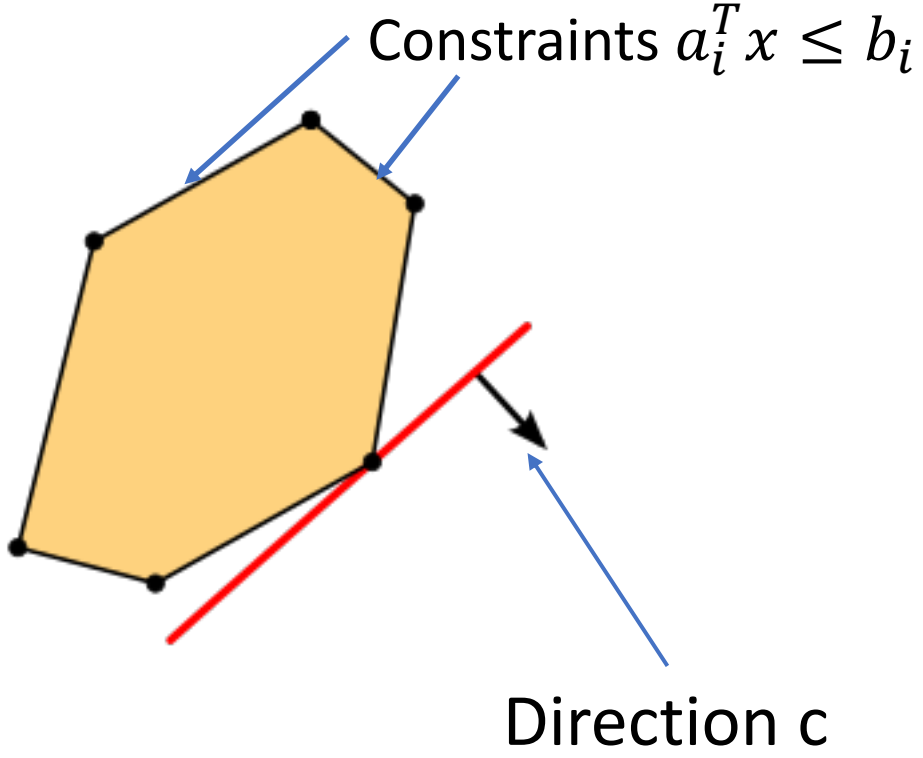
Linear and Integer Linear Programming

Linear programming



General form of LP

Find a vector	$\mathbf{x}$
that maximizes	$\mathbf{c}^T\mathbf{x}$
subject to	$A\mathbf{x} \leq \mathbf{b}$
and	$\mathbf{x} \geq \mathbf{0}$.



Linear programming

```
#include <stdio.h>          /* C input/output                       */
#include <stdlib.h>         /* C standard library                   */
#include <glpk.h>           /* GNU GLPK linear/mixed integer solver */

 /* declare variables */
  glp_prob *lp;
```

Linear programming

```
/* create problem */
 lp = glp_create_prob();
 glp_set_prob_name(lp, "short");
 glp_set_obj_dir(lp, GLP_MAX);
```

Linear programming

```
/* fill problem */
 glp_add_rows(lp, 2);
 glp_set_row_name(lp, 1, "p");
 glp_set_row_bnds(lp, 1, GLP_UP, 0.0, 1.0);
 glp_set_row_name(lp, 2, "q");
 glp_set_row_bnds(lp, 2, GLP_UP, 0.0, 2.0);

 glp_add_cols(lp, 2);
 glp_set_col_name(lp, 1, "x1");
 glp_set_col_bnds(lp, 1, GLP_LO, 0.0, 0.0);
 glp_set_obj_coef(lp, 1, 0.6);
 glp_set_col_name(lp, 2, "x2");
 glp_set_col_bnds(lp, 2, GLP_LO, 0.0, 0.0);
 glp_set_obj_coef(lp, 2, 0.5);
```

Linear programming

```
ia[1] = 1, ja[1] = 1, ar[1] = 1.0; /* a[1,1] = 1 */
ia[2] = 1, ja[2] = 2, ar[2] = 2.0; /* a[1,2] = 2 */
ia[3] = 2, ja[3] = 1, ar[3] = 3.0; /* a[2,1] = 3 */
ia[4] = 2, ja[4] = 2, ar[4] = 1.0; /* a[2,2] = 1 */

glp_load_matrix(lp, 4, ia, ja, ar);
```

Linear programming

```
/* solve problem */
glp_simplex(lp, NULL);

/* recover and display results */
z = glp_get_obj_val(lp);
x1 = glp_get_col_prim(lp, 1);
x2 = glp_get_col_prim(lp, 2);
printf("z = %g; x1 = %g; x2 = %g\n", z, x1, x2);

/* housekeeping */
glp_delete_prob(lp);
glp_free_env();
```

Linear programming

```
$ gcc glpkdemo.c -lglpk
$ ./a.out
GLPK Simplex Optimizer, v4.65
2 rows, 2 columns, 4 non-zeros
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (2)
*     2: obj =   4.600000000e-01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
z = 0.46; x1 = 0.6; x2 = 0.2
```

Integer Programming

Maximize

obj: x1 + 2 x2 + 3 x3 + x4

Subject To

c1: - x1 + x2 + x3 + 10 x4 <= 20

c2: x1 - 3 x2 + x3 <= 30

c3: x2 - 3.5 x4 = 0

Bounds

0 <= x1 <= 40

2 <= x4 <= 3

Integer

x4

Integer Programming

```
#include <stdlib.h>
#include <stdio.h>
#include <glpk.h>

glp_prob *mip = glp_create_prob();
glp_set_prob_name(mip, "sample");
glp_set_obj_dir(mip, GLP_MAX);
glp_add_rows(mip, 3);
glp_set_row_name(mip, 1, "c1");
glp_set_row_bnds(mip, 1, GLP_DB, 0.0, 20.0);
glp_set_row_name(mip, 2, "c2");
glp_set_row_bnds(mip, 2, GLP_DB, 0.0, 30.0);
glp_set_row_name(mip, 3, "c3");
glp_set_row_bnds(mip, 3, GLP_FX, 0.0, 0);
```

Integer Programming

```
glp_add_cols(mip, 4);
 glp_set_col_name(mip, 1, "x1");
 glp_set_col_bnds(mip, 1, GLP_DB, 0.0, 40.0);
 glp_set_obj_coef(mip, 1, 1.0);
 glp_set_col_name(mip, 2, "x2");
 glp_set_col_bnds(mip, 2, GLP_LO, 0.0, 0.0);
 glp_set_obj_coef(mip, 2, 2.0);
 glp_set_col_name(mip, 3, "x3");
 glp_set_col_bnds(mip, 3, GLP_LO, 0.0, 0.0);
 glp_set_obj_coef(mip, 3, 3.0);
 glp_set_col_name(mip, 4, "x4");
 glp_set_col_bnds(mip, 4, GLP_DB, 2.0, 3.0);
 glp_set_obj_coef(mip, 4, 1.0);
 glp_set_col_kind(mip, 4, GLP_IV);
```

Integer Programming

```
int ia[1+9], ja[1+9];
 double ar[1+9];
 ia[1]=1,ja[1]=1,ar[1]=-1;   // a[1,1] = -1
 ia[2]=1,ja[2]=2,ar[2]=1;    // a[1,2] = 1
 ia[3]=1,ja[3]=3,ar[3]=1;    // a[1,3] = 1
 ia[4]=1,ja[4]=4,ar[4]=10;   // a[1,4] = 10
 ia[5]=2,ja[5]=1,ar[5]=1;    // a[2,1] = 1
 ia[6]=2,ja[6]=2,ar[6]=-3;   // a[2,2] = -3
 ia[7]=2,ja[7]=3,ar[7]=1;    // a[2,3] = 1
 ia[8]=3,ja[8]=2,ar[8]=1;    // a[3,2] = 1
 ia[9]=3,ja[9]=4,ar[9]=-3.5; // a[3,4] = -3.5
 glp_load_matrix(mip, 9, ia, ja, ar);
```

Integer Programming

```
/* integer optimizer control parameters  */
glp_iocp parm;
 glp_init_iocp(&parm);
 parm.presolve = GLP_ON;

/* integer optimizer */
 int err = glp_intopt(mip, &parm);
```

Integer Programming

```
$ ./a.out
GLPK Integer Optimizer, v4.65
3 rows, 4 columns, 9 non-zeros
1 integer variable, none of which are binary
Preprocessing...
3 rows, 4 columns, 9 non-zeros
1 integer variable, none of which are binary
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+01  ratio =  1.000e+01
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 3
Solving LP relaxation...
GLPK Simplex Optimizer, v4.65
3 rows, 4 columns, 9 non-zeros
      0: obj =   2.300000000e+01 inf =   1.400e+01 (1)
      1: obj =   3.700000000e+01 inf =   0.000e+00 (0)
*     5: obj =   1.252083333e+02 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins...
Long-step dual simplex will be used
+     5: mip =     not found yet <=           +inf       (1; 0)
+     6: >>>>>   1.225000000e+02 <=   1.225000000e+02 < 0.1% (2; 0)
+     6: mip =   1.225000000e+02 <=     tree is empty   0.0% (0; 3)
INTEGER OPTIMAL SOLUTION FOUND
```

z = 122.5; x1 = 40; x2 = 10.5; x3 = 19.5, x4 = 3

Applications of Linear Programming

Maximum Flow Problem

- Given network (V,E), and a pair of source and sink vertices: (s, t)
- c_{uv} is the capacity of edge (u, v).
- Find the maximum flow from source to sink

- Let f_{uv} denote the optimal flow from vertex u to v and $g_{uv} = 1$ if there is and edge from u to v, 0 otherwise

- Objective: $Max_{f_{uv}} \sum_u f_{su} g_{su}$
- Subject to: $\sum_u f_{uv} g_{uv} = \sum_{u'} f_{vu'} g_{vu'} \ \forall v$ except s,t

$$0 \leq f_{uv} \leq c_{uv} \ \forall \ (u, u) \in E$$