Constraints from Sketchpad to Apple Auto Layout

Some research topics by my students and me over the years:

- a theory of hard and soft constraints
- constraint solvers
  (notable ones: DeltaBlue, Cassowary)
- constraint toolkits
- constraints in logic programming languages, constraint imperative languages
- recently: object constraint programming

Constraints for Layout and Interactivity

- Many layout problems can be expressed using constraints. Need both equalities and inequalities, and also preferences as well as requirements.
- Need to handle change over time (moving, resizing, etc.)
- We need to handle cycles (both simultaneous equalities and simultaneous inequalities).

Constraint Hierarchy Formalism

Allow both required and preferential constraints (hard and soft constraints). Benefits of a declarative theory: can argue convincingly that the solutions from a solver are correct (or not); criterion for deciding whether optimizations are correct

Important uses in layout:

- stability when moving a figure (use weak constraints that things stay where they are)
- handling user inputs that are outside the range of permitted values
- balancing conflicting desires (in layout and elsewhere)

Constraint Hierarchy Solutions

There can be an arbitrary number of constraint strengths.

In finding a solution, we *must* satisfy the required constraints. We try to satisfy the stronger non-required constraints in preference to the weaker ones.

required	$x + y = 10$
strong	$x = 8$
weak	$x = 0$
weak	$y = 0$

Solution is $\{x \mapsto 8, y \mapsto 2\}$

Error in Satisfying a Constraint

If we can't satisfy a preferential constraint exactly, we try to satisfy it as well as possible.

$$\begin{array}{rc} \text{required} & 10 \leq x \leq 20 \\ \text{weak} & x = 5 \end{array}$$

The error for $x = 5$ is $|x - 5|$.

The solution is $\{x \mapsto 10\}$.

Conflicting Constraints

Two conflicting constraints with the same strength:

$$\begin{array}{rl} \text{required} & x + y = 10 \\ \text{strong} & x = 0 \\ \text{strong} & y = 0 \end{array}$$

There are various ways of trading off constraints in this situation.

The Cassowary solver will find either
$\{x \mapsto 0, y \mapsto 10\}$ or
$\{x \mapsto 10, y \mapsto 0\}$.

A least-squares solver (e.g. QOCA) will find
$\{x \mapsto 5, y \mapsto 5\}$.

The Cassowary Constraint Solver

Alan Borning

this was joint work with:
Greg Badros (University of Washington,
now retired from Facebook)
Kim Marriott (Monash University)
Peter Stuckey (University of Melbourne)

Linear Programming

Linear programming is concerned with solving the following problem.

$x_1, \ldots, x_n$ are non-negative real-valued variables: $x_i \geq 0$ for $1 \leq i \leq n$.

There are m linear equality or inequality constraints over the x_i, each of the form:

$$\begin{array}{lrcl} & a_1x_1 + \ldots + a_nx_n & = & c, \\ & a_1x_1 + \ldots + a_nx_n & \leq & c, \\ \text{or} & a_1x_1 + \ldots + a_nx_n & \geq & c. \end{array}$$

Find values for the x_i that minimizes the value of the *objective function*

$$c + d_1x_1 + \ldots + d_nx_n$$

Government Health Warning

As you see, the subject of linear programming is surrounded by notational and terminological thickets. Both of these thorny defenses are lovingly cultivated by a coterie of stern acolytes who have devoted themselves to the field.

– *Numerical Recipes, page 424*

Simplex Algorithm — Executive Summary

1. Convert the problem to simplex form by using slack variables to convert inequalities to equalities. Example: replace $x \leq 10$ by $x + s = 10$. (Remember that both x and s must be non-negative.)

2. Find *a* solution to the constraints (not necessarily the optimal one). Finding this solution will involve putting the problem into a kind of normal form called *basic feasible solved form*.

3. Try to decrease the value of the objective function using a matrix operation on the solved form called a *pivot*.

4. Keep pivoting until the value of the objective function can be decreased no further.

Basic Feasible Solved Form

The problem is in *basic feasible solved form* if each equation is of the form

$$x_0 = c + a_1 x_1 + \ldots + a_n x_n$$

where the variable x_0 does not occur in any other equation or in the objective function, and c is non-negative.

The set of such equations is called the *tableau*.

The variable x_0 is *basic* and the other variables in the equation are *parameters*.

A problem in basic feasible solved form defines a *basic feasible solution*. We can read off this solution by setting each parametric variable to 0 and each basic variable to the value of the constant in the right-hand side.

Problems with Using Simplex

Incrementality:

- resolving the constraints for new input values, e.g. after moving the mouse
- incremental addition and deletion of constraints

Other issues:

- variables that may take on negative values
- non-linear objective function

Allowing Unrestricted Variables

The simplex algorithm imposes a constraint $x \geq 0$ on all its variables.

To escape this restriction efficiently, we use an *augmented simplex algorithm* with two tableaux: C_U and C_S.

- C_U: the unrestricted variable tableau
- C_S: the simplex tableau. Only restricted variables are allowed in C_S.

We use the simplex algorithm to optimize the objective function, based only on the equations in C_S.

We will still be able to read off solutions from the two tableaux as before.

Augmented Simplex Form Example

Consider the constraints

$$\begin{gathered}-5 \leq x \leq 20 \\ y + 10 = x\end{gathered}$$

Add slack variables:

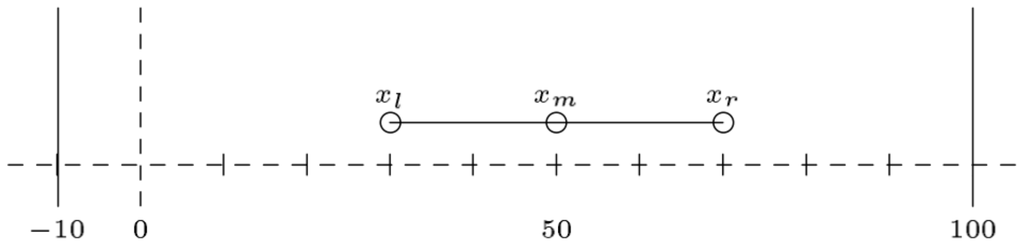
$$\begin{aligned}-5 + s_1 &= x \\ x + s_2 &= 20 \\ y + 10 &= x\end{aligned}$$

In augmented simplex form:

$$\begin{array}{rcr} y & = & -15 + s_1 \\ x & = & -5 + s_1 \\ \hline s_2 & = & 25 - s_1 \end{array}$$

Solution: $\{y \mapsto -15, x \mapsto -5, s_2 \mapsto 25, s_1 \mapsto 0\}$.

Handling Non-Required Constraints



Suppose the user wishes to edit x_m in the diagram and have x_l and x_r weakly stay where they are.

This adds the non-required constraints x_m *edit*, x_l *stay*, and x_r *stay*.

Suppose we are trying to move x_m to position 50, and that currently $x_l = 30$, $x_m = 45$, and $x_r = 60$.

Handling Non-Required Constraints (2)

The complete set of constraints:

required	$2x_m = x_l + x_r$
required	$x_l + 10 \leq x_r$
required	$x_r \leq 100$
required	$-100 \leq x_l$
strong	$x_m = 50$
weak	$x_l = 30$
weak	$x_r = 60$

Objective Function

We combine the errors for each non-required constraint with a weight.

Objective function:

$$s|x_m - 50| + w|x_l - 30| + w|x_r - 60|$$

where s and w are weights so that the strong constraint is always strictly more important than solving any combination of weak constraints. This lets us find a locally-error-better solution.

This objective function is non-linear.

Symbolic Weights

$$s|x_m - 50| + w|x_l - 30| + w|x_r - 60|$$

To avoid problems with choosing too small an s, rather than a real number we use a symbolic weight and a lexicographic ordering for comparing values:

$$[1,0]\times|x_m-50|+[0,1]\times|x_l-30|+[0,1]\times|x_r-60|$$

These symbolic weights are instances of a class SymbolicWeight that understands the usual arithmetic messages (+, *, =, $\leq$, etc), so that we can compare symbolic weights, add two symbolic weights, multiply a symbolic weight by a real, and so forth.

Error Variables

The edit and the stay constraints will be represented as equations of the form

$$v = \alpha + \delta_v^+ - \delta_v^-$$

where δ_v^+ and δ_v^- are non-negative variables representing the deviation of v from the desired value α.

If the constraint is satisfied both δ_v^+ and δ_v^- will be 0. Otherwise δ_v^+ will be positive and δ_v^- will be 0 if v is too big, or vice versa if v is too small.

Non-Required Constraints for Augmented Simplex

The constraints strong $x_m = 50$, weak $x_l = 30$, and weak $x_r = 60$ become:

$$\begin{array}{rcl} x_m & = & 50 + \delta^+_{x_m} - \delta^-_{x_m} \\ x_l & = & 30 + \delta^+_{x_l} - \delta^-_{x_l} \\ x_r & = & 60 + \delta^+_{x_r} - \delta^-_{x_r} \\ \\ 0 & \leq & \delta^+_{x_m}, \delta^-_{x_m}, \delta^+_{x_l}, \delta^-_{x_l}, \delta^+_{x_r}, \delta^-_{x_r} \end{array}$$

The objective function is

minimize

$$\begin{array}{rlcl} & [1,0]\delta^+_{x_m} & + & [1,0]\delta^-_{x_m} \\ + & [0,1]\delta^+_{x_l} & + & [0,1]\delta^-_{x_l} \\ + & [0,1]\delta^+_{x_r} & + & [0,1]\delta^-_{x_r} \end{array}$$

An Optimal Solution

minimize $[0,10] + [1,2]\delta^{+}_{x_m} + [1,-2]\delta^{-}_{x_m} + [0,2]\delta^{-}_{x_l} + [0,2]\delta^{-}_{x_r}$ subject to

$$\begin{array}{rcrrrrrr}
x_m & = & 50 & +\delta^{+}_{x_m} & -\delta^{-}_{x_m} & & & \\
x_l & = & 30 & & & +\delta^{+}_{x_l} & -\delta^{-}_{x_l} & \\
x_r & = & 70 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & -\delta^{+}_{x_l} & +\delta^{-}_{x_l} & \\
\hline
s_1 & = & 30 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & -2\delta^{+}_{x_l} & +2\delta^{-}_{x_l} & \\
s_3 & = & 30 & -2\delta^{+}_{x_m} & +2\delta^{-}_{x_m} & +\delta^{+}_{x_l} & -\delta^{-}_{x_l} & \\
\delta^{+}_{x_r} & = & 10 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & -\delta^{+}_{x_l} & +\delta^{-}_{x_l} & +\delta^{-}_{x_r} \\
s_2 & = & 40 & & & +\delta^{+}_{x_l} & -\delta^{-}_{x_l} &
\end{array}$$

Old positions: $\{x_l \mapsto 30, x_m \mapsto 45, x_r \mapsto 60\}$
New positions: $\{x_l \mapsto 30, x_m \mapsto 50, x_r \mapsto 70\}$

The weak stay on x_r is not satisfied.

Incrementality: Resolving the Optimization Problem

Suppose the user is moving some part of a constrained figure with the mouse. Each time the screen is refreshed we need to solve the constraints again — so this needs to be fast!

The only difference between the successive problems will be in the values of the constants in the edit and stay constraints.

Suppose the user is editing x_m. The user moves the mouse to $x = 60$.

We wish to solve a new problem, with new values for the edit and stay constraints:

$$\begin{array}{rr} \text{strong} & x_m = 60 \\ \text{weak} & x_l = 30 \\ \text{weak} & x_r = 70 \end{array}$$

Incrementality: Resolving the Optimization Problem (2)

We want to modify the current tableau rather than starting from scratch.

Steps:

1. Modify the tableau to reflect the new constants for the stay constraints. No re-optimization required.

2. Modify the tableau to reflect the new constants for the edit constraints. May need to re-optimize.

Updating the Stay Constraints

Set certain constants in the tableau to 0.

In the example, we set the constant of the equation for $\delta^+_{x_r}$ to 0.

Result:

minimize $[0,0]+[1,2]\delta^+_{x_m}+[1,-2]\delta^-_{x_m}+[0,2]\delta^-_{x_l}+[0,2]\delta^-_{x_r}$ subject to

$$\begin{array}{rclrrrrr}
x_m & = & 50 & +\delta^+_{x_m} & -\delta^-_{x_m} & & & \\
x_l & = & 30 & & & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
x_r & = & 70 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & \\
\hline
s_1 & = & 30 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -2\delta^+_{x_l} & +2\delta^-_{x_l} & \\
s_3 & = & 30 & -2\delta^+_{x_m} & +2\delta^-_{x_m} & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
\delta^+_{x_r} & = & 0 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & +\delta^-_{x_r} \\
s_2 & = & 40 & & & +\delta^+_{x_l} & -\delta^-_{x_l} &
\end{array}$$

Updating the Edit Constraints

Increment constants for certain rows in the tableau.

Pick up x_m with the mouse and move by 10 (from 50 to 60): add 10 times the coefficient of $\delta^+_{x_m}$ to the constant part of every row in which it occurs.

The modified tableau is

minimize $[0,20] + [1,2]\delta^+_{x_m} + [1,-2]\delta^-_{x_m} + [0,2]\delta^-_{x_l} + [0,2]\delta^-_{x_r}$ subject to

$$
\begin{array}{rclrrrrr}
x_m & = & 60 & +\delta^+_{x_m} & -\delta^-_{x_m} & & & \\
x_l & = & 30 & & & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
x_r & = & 90 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & \\
\hline
s_1 & = & 50 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -2\delta^+_{x_l} & +2\delta^-_{x_l} & \\
s_3 & = & 10 & -2\delta^+_{x_m} & +2\delta^-_{x_m} & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
\delta^+_{x_r} & = & 20 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & +\delta^-_{x_r} \\
s_2 & = & 40 & & & +\delta^+_{x_l} & -\delta^-_{x_l} &
\end{array}
$$

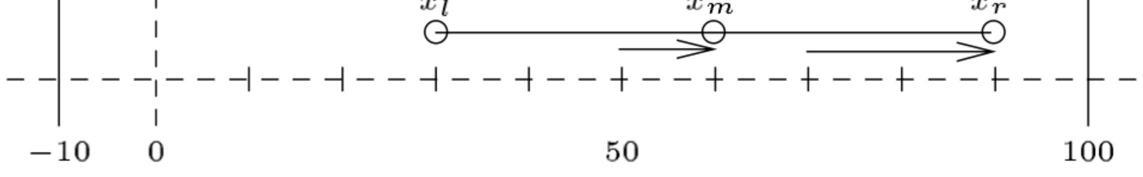
New Solution

The simplex tableau is feasible and already in optimal form, and so we have incrementally resolved the problem by simply modifying constants in the tableaux.

New solution

$$\{x_l \mapsto 30, x_m \mapsto 60, x_r \mapsto 90\}$$

Sliding the midpoint rightwards has caused the right point to slide rightwards as well, but twice as far.



Another Edit

Move x_m from 60 to 90. The updated tableau is

minimize $[0,60] + [1,2]\delta^+_{x_m} + [1,-2]\delta^-_{x_m} + [0,2]\delta^-_{x_l} + [0,2]\delta^-_{x_r}$ subject to

$$
\begin{array}{rcrrrrrr}
x_m & = & 90 & +\delta^+_{x_m} & -\delta^-_{x_m} & & & \\
x_l & = & 30 & & & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
x_r & = & 150 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & \\
\hline
s_1 & = & 110 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -2\delta^+_{x_l} & +2\delta^-_{x_l} & \\
s_3 & = & -50 & -2\delta^+_{x_m} & +2\delta^-_{x_m} & +\delta^+_{x_l} & -\delta^-_{x_l} & \\
\delta^+_{x_r} & = & 60 & +2\delta^+_{x_m} & -2\delta^-_{x_m} & -\delta^+_{x_l} & +\delta^-_{x_l} & +\delta^-_{x_r} \\
s_2 & = & 40 & & & +\delta^+_{x_l} & -\delta^-_{x_l} &
\end{array}
$$

The tableau is no longer in basic feasible solved form (s_2 is negative).

Intuition: x_r should have stopped at the virtual wall created by the $x_r \leq 100$ constraint, but in these tableaux has crashed on through.

Re-optimizing

In general, after updating the constants for the edit constraints, the simplex tableau C_S may no longer be in basic feasible solved form, since some of the constants may be negative.

In this case use the *dual simplex algorithm* to re-optimize.

The dual simplex algorithm starts from an optimal and infeasible solution. Each pivot moves toward feasibility, all the while maintaining optimality. We stop when we reach feasibility.

(Contrast this with the primal simplex algorithm, where we start from a feasible solution. Each pivot moves toward optimality, all the while maintaining feasibility. We stop when we reach optimality)

After Pivoting

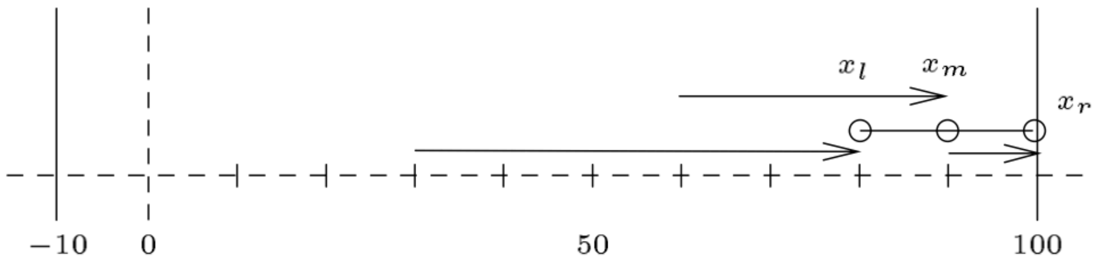
We perform a dual simplex pivot. Result:

minimize $[0,60] + [1,2]\delta^{+}_{x_m} + [1,-2]\delta^{-}_{x_m} + [0,2]\delta^{-}_{x_l} + [0,2]\delta^{-}_{x_r}$ subject to

$$\begin{array}{rcrrrrrr}
x_m & = & 90 & & +\delta^{+}_{x_m} & -\delta^{-}_{x_m} & & \\
x_l & = & 80 & +s_3 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & & \\
x_r & = & 100 & -s_3 & & & & \\
\hline
s_1 & = & 10 & -2s_3 & -2\delta^{+}_{x_m} & +2\delta^{-}_{x_m} & & \\
\delta^{+}_{x_l} & = & 50 & +s_3 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & +\delta^{-}_{x_l} & \\
\delta^{+}_{x_r} & = & 10 & -s_3 & & & & +\delta^{-}_{x_r} \\
s_2 & = & 90 & +s_3 & +2\delta^{+}_{x_m} & -2\delta^{-}_{x_m} & &
\end{array}$$

The tableau is now feasible and optimal.
Solution: $\{x_l \mapsto 80, x_m \mapsto 90, x_r \mapsto 100\}$.

After Pivoting (2)



We slide the midpoint to the right.

Eventually the rightmost point hits the wall and the left point then starts moving to satisfy the constraints.

Benefits of Dual Simplex

Pivoting occurred when the right point x_r came up against a barrier.

If we picked up the midpoint x_m with the mouse and smoothly slid it rightwards, 1 pixel every screen re-fresh, only one pivot would be required in moving from 50 to 95.

This illustrates why the dual optimization is well suited to this problem and leads to efficient resolving.

Incrementality: Adding and Deleting Constraints

We can incrementally add a constraint to an existing tableau, or delete an arbitrary constraint from a tableau — no need to start from scratch.

Important use: adding an edit constraint to move a part, then deleting that constraint when done moving.

The Cassowary solver includes cool algorithms for this — see the journal article for details!