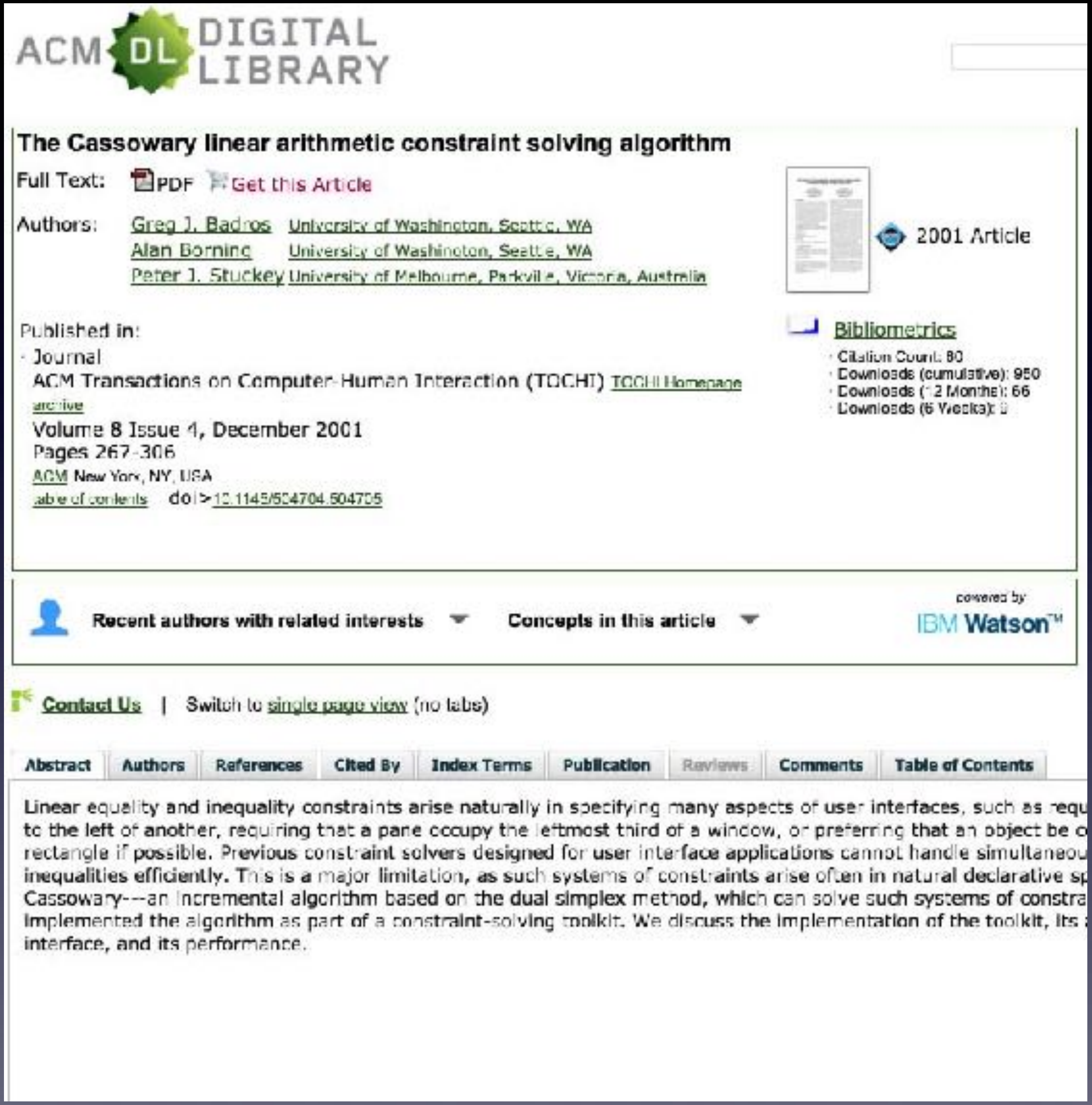
iOS Layout

Cassoway Algorithm

http://overconstrained.io/


ACM DL DIGITAL LIBRARY

The Cassowary linear arithmetic constraint solving algorithm

Full Text: PDF Get this Article

Authors: Greg J. Badros University of Washington, Seattle, WA
Alan Borning University of Washington, Seattle, WA
Peter J. Stuckey University of Melbourne, Parkville, Victoria, Australia

2001 Article

Published in:
· Journal
ACM Transactions on Computer-Human Interaction (TOCHI) TOCHI Homepage archive
Volume 8 Issue 4, December 2001
Pages 267-306
ACM New York, NY, USA
table of contents doi>10.1145/504704.504705

Bibliometrics
· Citation Count: 80
· Downloads (cumulative): 950
· Downloads (12 Months): 66
· Downloads (6 Weeks): 9

Recent authors with related interests Concepts in this article powered by IBM Watson™

Contact Us | Switch to single page view (no tabs)

Abstract | Authors | References | Cited By | Index Terms | Publication | Reviews | Comments | Table of Contents

Linear equality and inequality constraints arise naturally in specifying many aspects of user interfaces, such as requ
to the left of another, requiring that a pane occupy the leftmost third of a window, or preferring that an object be c
rectangle if possible. Previous constraint solvers designed for user interface applications cannot handle simultaneou
inequalities efficiently. This is a major limitation, as such systems of constraints arise often in natural declarative sp
Cassowary---an incremental algorithm based on the dual simplex method, which can solve such systems of constra
implemented the algorithm as part of a constraint-solving toolkit. We discuss the implementation of the toolkit, its
interface, and its performance.

Anatomy of a constraint

- 2 views
- 2 view attributes
- Constant
- Multiplier (Dimension and center only)
- Relation (>=, <=, =)

Constraints in Interface Builder

- Auto layout needs to compute the **position** (x,y) and **size** (width, height) of each view
- Provide any 2 horizontal axis constraints to a guide (or view) with known position (and size): **leading, trailing, centerX, width** [aspect ratio]
- Provide any 2 vertical axis constraints to a guide (or view) with known position (and size): **top, bottom, centerY, height** [aspect ratio], **firstBaseline, lastBaseline**
- exception**:** views **intrinsic sizes** only need a position

- Creating constraints with the document outline
  - Ctrl + drag to create a constraint
  - Shift + click for multiple constraints
  - Option + click for alternate constraints (margins / guide)

- Editing constraints with the size inspector
  - constants
  - inequality constraints
  - proportional constraints with center multipliers
  - constraints you can't create in the document outline (baseline to edge, center to edge)
  - fixing margin mistakes
  - reversing order to fix negative constants & reciprocal

- Prefer UIStackView to constraints, then add constraints after
    - Alignment is the axis perpendicular to the axis of the arranged subviews
    - Distribution is the axis of the arranged subviews

Constraint outlets

- Ctrl + Drag to create an outlet to a constraint
- You can modify (and animate) the constant of a constraint
- You can set a constraint isActive to dynamically enable or disable it
- Asynch recalculation of frames before the next redraw with setNeedsLayout
- Synchronous recalculation of frames with layoutIfNeeded

Autolayout in Code

- NSLayoutConstraint
    - visual format (don't use it)
- NSLayoutAnchor

Making a radial menu with NSLayoutAnchor

Not So Autolayout

- In a UIViewController calculation of frames is done once **viewDidLayoutSubviews** is called. All size dependent code goes here. Never put sizing code in **ViewDidLoad**.

- In a **UIView** your view's frame has been calculated once **layoutSubviews** has been called. This is the place to adjust anything not managed by autolayout (ie **CALayer**)

- You can also draw manually in **UIView**.**drawRect**

Manual Layout with CAShapeLayer: RadialProgressView

UICollectionViewFlowLayout