1. Introduction and Summary

This report examines the antecedents of the following key paper by George Mason University (GMU) Professor Edward Wegman and GMU Assistant Research Professor Yasmin Said.


Wegman is best known as the lead author of the “hockey stick” report to the U.S. Congress, commissioned by Rep. Joe Barton in 2005, and released in July 2006. Said was a co-author of that report, as was David Scott of Rice University; the three are also the editors of the WIREs Comp Stat journal. Said is a protégé of Wegman, having received her PhD under his direction in 2005. She went on to become one of Wegman’s most frequent co-authors.

“Roadmap for Optimization” was the first article in the very first issue of WIREs CompStat, and one of two major overview articles by these authors. (The other one is Wegman and Said 2011, “Color Theory and Design”, for which I have previously performed a similar analysis).

The abstract reads:

This article focuses broadly on the area known as optimization. The intent is to provide in broad brush strokes a perspective on the area in order to orient the reader to more detailed treatments of specific subdisciplines of optimization throughout WIREs: Computational Statistics. In this article we provide background on mathematical programming, Lagrange multipliers, Karush-Kuhn-Tucker Conditions, numerical optimization methods, linear programming, dynamic programming, the calculus of variations, and metaheuristic algorithms.

Thus, this article was conceived as a keystone in realizing the “serial encyclopedia” vision of WIREs Comp Stat.

As can be seen in the following table, apparent unattributed antecedents have been identified in each of the sections corresponding to the aforementioned subjects. For the most part, the antecedents’ use appears sequential, and shows little of the interspersing of sources found in “Color Theory and Design”.

In all, 15 online sources have been identified as likely antecedents; these include 13 Wikipedia articles and two others (one from Prof Tom Ferguson and one from Wolfram MathWorld).

Section 2 (p. 3-5) of this report breaks down Said and Wegman 2009 page by page and gives identified antecedents, if any, for each sub-section, paragraph by paragraph. This section also contains a brief selection of identified errors, most of which have been introduced through various changes to the antecedents, reflecting apparent mistranscriptions or misunderstandings of subject area details. Some key omissions are also identified.

Section 3 (p. 6-30) begins with a detailed breakdown of the antecedents paragraph by paragraph. That is followed by a complete textual side-by-side comparison of Said and Wegman and its identified antecedents, with identical and trivially similar text highlighted.

Finally, section 4 (p. 31-33) has a brief analysis of Said and Wegman’s references and suggested reading list. The seven references, including two for different editions of Bellman’s Dynamic Programming, can all be found in the corresponding Wikipedia antecedents. Similarly, the extensive “Further Reading” list also appears to be derived from the various antecedents.
<table>
<thead>
<tr>
<th>Table 1: Antecedents of Said and Wegman 2009</th>
<th>Said &amp; Wegman (2009)</th>
<th>Source (with hyperlink and date of version analyzed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7. Tom Ferguson – Linear Programming: A Concise Introduction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13. Wolfram MathWorld – Calculus of Variations</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Detailed antecedents of Said and Wegman 2009

<table>
<thead>
<tr>
<th>Sect.</th>
<th>Page</th>
<th>Section title</th>
<th>Para</th>
<th>Source (with hyperlink)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>INTRODUCTION</td>
<td>1</td>
<td>No antecedent found</td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td>INTRODUCTION (cont.)</td>
<td>2-5</td>
<td>Wikipedia – Mathematical Optimization (Jan. 2009)</td>
</tr>
<tr>
<td></td>
<td>4-5</td>
<td>LAGRANGE MULTIPLIERS (cont.)</td>
<td>2-11</td>
<td>Wikipedia – Karush-Kuhn-Tucker conditions (Jan. 2009)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>NUMERICAL OPTIMIZATION METHODS - Newton’s Method</td>
<td>1-2</td>
<td>No antecedent found</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>NUMERICAL OPTIMIZATION METHODS - Gradient Descent</td>
<td>3</td>
<td>Wikipedia – Gradient Descent (Nov. 2008)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>NUMERICAL OPTIMIZATION METHODS - Conjugate Gradient Methods</td>
<td>4-6</td>
<td>Wikipedia – Conjugate Gradient Method (Jan. 2009)</td>
</tr>
<tr>
<td>4</td>
<td>5-6</td>
<td>LINEAR PROGRAMMING</td>
<td>1, 2*, 3</td>
<td>Wikipedia – Linear Programming (Jan. 2009)</td>
</tr>
<tr>
<td>6</td>
<td>5-6</td>
<td>LINEAR PROGRAMMING</td>
<td>2*</td>
<td>Tom Ferguson – Linear Programming: A Concise Introduction</td>
</tr>
<tr>
<td>6-7</td>
<td></td>
<td>LINEAR PROGRAMMING - Simplex Algorithm</td>
<td>4-6</td>
<td>Wikipedia – Simplex Algorithm (Jan. 2009)</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>LINEAR PROGRAMMING - Interior Point Methods and Karmarkar’s Algorithm</td>
<td>7*, 8-9</td>
<td>Wikipedia – Karmarkar's Algorithm (Jan. 2009)</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>LINEAR PROGRAMMING - Interior Point Methods and Karmarkar’s Algorithm</td>
<td>7*</td>
<td>Wikipedia – Interior Point Methods (Dec. 2008)</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>DYNAMIC PROGRAMMING (p. 7)</td>
<td>1-2</td>
<td>Wikipedia – Dynamic Programming (Jan. 2009)</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>CALCULUS OF VARIATIONS (p. 8)</td>
<td>1-2</td>
<td>Wikipedia – Calculus of Variations (Dec. 2008)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>CALCULUS OF VARIATIONS (p. 8)</td>
<td>3*, 6*</td>
<td>Wolfram MathWorld – Calculus of Variations</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>CALCULUS OF VARIATIONS (p. 8)</td>
<td>3*, 4-5, 6*</td>
<td>Wiki. – Fundamental Lemma of Calculus of Variations (Jan. 2009)</td>
</tr>
<tr>
<td>8-9</td>
<td></td>
<td>METAHEURISTIC ALGORITHMS – Evolutionary Algorithms, Genetic Algorithms</td>
<td>1-8</td>
<td>No antecedent found</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>METAHEURISTIC ALG. - Simulated Annealing</td>
<td>9, 10-11</td>
<td>Wikipedia – Simulated Annealing (Jan. 2009)</td>
</tr>
<tr>
<td>9-10</td>
<td></td>
<td>METAHEURISTIC ALG. – Tabu Search</td>
<td>12-13</td>
<td>No antecedent found</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>CONCLUSION</td>
<td>13</td>
<td>No antecedent found</td>
</tr>
</tbody>
</table>
The following table enumerates a selection of identified errors, all of which appear to have been introduced in the changing of antecedent material. Each error is excerpted in the first column, with the corresponding antecedent and an explanation of the error, and a pointer to the page number corresponding to the changing of antecedent material. Each error is excerpted in the first column, with the corresponding antecedent and an explanation of the error, and a pointer to the page number corresponding to the changing of antecedent material. Each error is excerpted in the first column, with the corresponding antecedent and an explanation of the error, and a pointer to the page number corresponding to the changing of antecedent material.

### Passage containing error

In this article we provide background on mathematical programming, Lagrange multipliers

The simplest case of mathematical optimization is mathematical programming. Mathematical programming considers as its objective function a scalar real-valued function of real variables.

The regularity conditions for a minimum point to satisfy the Karush–Kuhn–Tucker conditions are given below.

The conjugate gradient method is a recursive numerical method.

Two positive vectors $u$ and $v$ are conjugate with respect to $A$ if $u^TAv = 0$.

The hyperplane corresponding to the largest $v$ that still intersects the feasible region will intersect a vertex, a whole edge, or a face of the polytope.

… there are $d$ variables and $m$ constraints, not counting the non-negativity constraints.

… the simplex method visits all $2^d$ vertices before arriving at the optimal vertex.

2. Optimally solve the subproblems using the three step process iteratively.

$$\frac{df}{dy} - \frac{d}{dx} \left( \frac{df}{dy} \right) = 0.$$  

---

<table>
<thead>
<tr>
<th>Passage containing error</th>
<th>Sec</th>
<th>Pg</th>
<th>Antecedent</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this article we provide background on mathematical programming, Lagrange multipliers</td>
<td>Ab</td>
<td>1</td>
<td>N/A</td>
<td>Said and Wegman treat “mathematical programming” as a subtype of optimization, but it is used here as a synonym for optimization. The current version of Wikipedia reads: “[M]athematical optimization (alternatively, optimization or mathematical programming) …”</td>
</tr>
<tr>
<td>The simplest case of mathematical optimization is mathematical programming. Mathematical programming considers as its objective function a scalar real-valued function of real variables.</td>
<td>1</td>
<td>3</td>
<td>In mathematics, the simplest case of optimization, or mathematical programming, refers to the study of problems … [involving] … a scalar real valued objective function</td>
<td>The original makes clear that only one of the regularity conditions is necessary for $x^* \in \mathbb{KKT}$, but S&amp;W are ambiguous at best.</td>
</tr>
<tr>
<td>The regularity conditions for a minimum point to satisfy the Karush–Kuhn–Tucker conditions are given below.</td>
<td>2</td>
<td>4</td>
<td>In order for a minimum point $x^*$ be KKT, it should satisfy some regularity condition, the most used ones are listed below:</td>
<td>S &amp; W use “iterative” and “recursive” as synonyms (see below for a reverse example).</td>
</tr>
<tr>
<td>The conjugate gradient method is a recursive numerical method.</td>
<td>3</td>
<td>5</td>
<td>The conjugate gradient method is an iterative method …</td>
<td>Non-zero has been translated into “positive”.</td>
</tr>
<tr>
<td>Two positive vectors $u$ and $v$ are conjugate with respect to $A$ if $u^TAv = 0$.</td>
<td>3</td>
<td>5</td>
<td>We say that two non-zero vectors $u$ and $v$ are conjugate (with respect to $A$) if $u^TAv = 0$.</td>
<td>Here “intersect” is treated as a synonym for “just graze” (i.e. touching the surface only).</td>
</tr>
<tr>
<td>The hyperplane corresponding to the largest $v$ that still intersects the feasible region will intersect a vertex, a whole edge, or a face of the polytope.</td>
<td>4</td>
<td>6</td>
<td>the last hyperplane to intersect the feasible region will either just graze a vertex of the polytope, or a whole edge or face.</td>
<td>This appears to be an omission to change “$n$” to “$d$”, unlike the other 9 cases in this section.</td>
</tr>
<tr>
<td>… there are $d$ variables and $m$ constraints, not counting the non-negativity constraints.</td>
<td>4</td>
<td>6</td>
<td>… there are $n$ variables and $m$ constraints, not counting the $n$ nonnegativity constraints</td>
<td>The exponent notation was lost, perhaps during copying or changing the “$n$” to ”$d$”.</td>
</tr>
<tr>
<td>… the simplex method visits all $2^d$ vertices before arriving at the optimal vertex.</td>
<td>4</td>
<td>7</td>
<td>… the simplex method … visits all $2^n$ vertices before arriving at the optimal vertex</td>
<td>The clearly recursive step is recast as “iterative”.</td>
</tr>
<tr>
<td>2. Optimally solve the subproblems using the three step process iteratively.</td>
<td>5</td>
<td>7</td>
<td>2. Solve these problems optimally using this three-step process recursively.</td>
<td>The Euler-Lagrange equation is mistranscribed.</td>
</tr>
<tr>
<td>$$\frac{df}{dy} - \frac{d}{dx} \left( \frac{df}{dy} \right) = 0.$$</td>
<td>6</td>
<td>8</td>
<td>$\frac{df}{dy} - \frac{d}{dx} \left( \frac{df}{dy} \right) = 0.$</td>
<td></td>
</tr>
</tbody>
</table>
The above table shows errors that can generally be described as mistranscriptions (e.g. $2^n$ vs $2d$) or as inappropriate phrasing (e.g. treating “recursive” and “iterative” as synonyms).

A third class of errors arise from omission, where key information has simply been left out. These can often be identified in side-by-side comparison by strikeout in the antecedent. The following table lists the three most egregious examples.

<table>
<thead>
<tr>
<th>Passage containing error</th>
<th>Sec</th>
<th>Pg</th>
<th>Antecedent</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppose $x_0$ is a starting point in the neighborhood of $x^*$. Then $\varphi$ decreases from $x_0$ at the maximum rate in the direction of the negative gradient of $\varphi(x_0)$.</td>
<td>3</td>
<td>5</td>
<td>then $F(x)$ decreases <strong>fastest</strong> if one goes from $a$ in the direction of the negative gradient of $F$ at $a$: $\nabla F(a)$. It follows that, if $b = a - \gamma \nabla F(a)$ for $\gamma &gt; 0$ a small enough number, then $F(a) &gt; F(b)$. With this observation in mind, one starts with a guess $x_0$ for a local minimum of $F$, and considers the sequence $x_0, x_1, x_2, \ldots$ such that $x_{n+1} = x_n - \gamma_n \nabla F(x_n)$, $n \geq 0$.</td>
<td><strong>There are a few omissions here, most notably the failure to specify $\nabla$ as the notation for gradient.</strong></td>
</tr>
</tbody>
</table>
| **Affine-Scaling Algorithm**
**Input:** $A, b, c$, stopping criterion, $\gamma$ | 4   | 7  | **Algorithm** **Affine-Scaling**
**Input:** $A, b, c$, $x^0$, stopping criterion, $\gamma$ |
| $k \leftarrow 0$
$\text{Do while stopping criterion is not satisfied}$
$k^k \leftarrow b - Ax^k$
| 4   | 7  | **During each step of the algorithm, the variable that will eventually represent the minimum is replaced by a random solution that is chosen according to a temperature parameter, } $T$. As the temperature of the system decreases, the probability of higher temperature values replacing the minimum decreases, but it is always non-zero. | **By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random “nearby” solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter $T$ (called the temperature), that is gradually decreased during the process.** |
| During each step of the algorithm, the variable that will eventually represent the minimum is replaced by a random solution that is chosen according to a temperature parameter, $T$. As the temperature of the system decreases, the probability of higher temperature values replacing the minimum decreases, but it is always non-zero. | 4   | 7  | **By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random “nearby” solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter $T$ (called the temperature), that is gradually decreased during the process.** |
| Said and Wegman’s version of Karmarkar’s algorithm leaves out a crucial input, namely the initial “guess”, $x^0$. | 26  | | **Said and Wegman’s explanation of simulated annealing omits key aspects of the approach, and implies that at each step a candidate solution is chosen randomly by sole reference to the global “temperature” parameter.** |
| Said and Wegman’s version of Karmarkar’s algorithm leaves out a crucial input, namely the initial “guess”, $x^0$. | 26  | | |
3. Detailed comparison of Roadmap for Optimization and antecedents

Colored regular font indicates substantially close wording between the article and its antecedent sources (identical in cyan, slight variation in yellow). Italic represent paraphrased sections. Bold represents significant departures of Wegman & Said 2011 from sources. To avoid clutter, function symbols and index subscripts have been shown as identical, where the mathematical meaning is not changed (e.g. \( \phi \) for \( f \)). Paragraphs have been reformatted for ease of comparison. Passages with changes that introduce various issues have been underlined (as enumerated in section 2).

Mathematical notation

Equations and expressions differing only in choice of function symbols and subscript letters have been shown as identical.

Consider this statement and expression from Roadmap (Linear Programming, p.5):

\[
\text{Similarly, constraints of the form } \sum_{j=1}^{d} a_{ij} x_{j} \geq b_{i} \text{ can be changed to } \\
\sum_{j=1}^{d} (-a_{ij}) x_{j} \leq -b_{i}.
\]

The corresponding summation expressions in the Wikipedia statement would be considered identical, despite the change in Roadmap of the upper bound from \( n \) to \( d \).

\[
\text{Similarly, constraints of the form } \sum_{j=1}^{n} a_{ij} x_{j} \geq b_{i} \text{ can be changed into the form } \\
\sum_{j=1}^{n} (-a_{ij}) x_{j} \leq -b_{i}.
\]

Here are more complex examples drawn from the section on Calculus of Variations (p. 8).

\[
\text{... stationary values of an integral of the form} \\
\mathcal{L}(f) = \int_{a}^{b} f(y, \dot{y}, x) dx:
\]

\[
\mathcal{L}(f) \text{ has an extremum only if the Euler–Lagrange differential equation is satisfied, i.e., if} \\
\frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{y}} \right) = 0.
\]

Here is the corresponding passage from Wolfram MathWorld:

\[
\text{... stationary values of integrals of the form} \\
I = \int_{a}^{b} f(y, \dot{y}, x) dx
\]

\[
I \text{ has an extremum only if the Euler-Lagrange differential equation is satisfied, i.e., if} \\
\frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{y}} \right) = 0.
\]

In this case, the RHS of the first equation is identical, but Roadmap’s more explicit “function-of-function” notation (i.e. \( L(f) \)) has been highlighted in yellow to show a trivial (i.e. non-semantic) change. The Euler-Lagrange equation has also been highlighted in yellow and underlined to show a mistranscription error in Roadmap, as previously explained in section 2; otherwise, it would have been identical.

The two equations are cited again (somewhat redundantly) a few paragraphs later, retaining the same formatting.

This lemma is used to prove that the extremals of \( \mathcal{L}(f) = \int_{a}^{b} f(y, \dot{y}, x) dx \) are weak solutions of the Euler–Lagrange equation:

\[
\frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{y}} \right) = 0.
\]

This time, though, the source is Wikipedia, which uses different notation.

This lemma is used to prove that extrema of the functional

\[
J[L(t, y, \dot{y})] = \int_{t_0}^{t_1} L(t, y, \dot{y}) dt
\]

are weak solutions of the Euler-Lagrange equation

\[
\frac{\partial L}{\partial y} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{y}} \right) = 0.
\]

PlanetMath has yet a third way of showing these.

\[
I(q) = \int_{a}^{b} L(t, q(t), \dot{q}(t)) dt
\]

\[
\frac{\partial}{\partial q} L - \frac{d}{dt} \left( \frac{\partial}{\partial \dot{q}} L \right) = 0.
\]
**Section 1 – INTRODUCTION (p. 3-4)**

**Para 1 – No antecedent found**

Optimization is a simple concept that involves a quantity that will be either minimized or maximized. In trying to optimize this quantity, certain parameters, sometimes called decision variables, are subject to constraints. In general, an optimization problem is comprised of three parts: the objective function, variables, and constraints. The objective function is the function to be optimized. The variables or parameters involved define the model. Finally, constraints determine the range of allowed values for the unknowns.

As taught in calculus, stationary points of smooth functions are points at which the derivative is equal to zero. In practical terms, these points represent maxima, minima, or saddle points. According to Weierstrass theorem, if a function has a finite domain, a maximum and a minimum will exist. The goal of optimization is to find these points if they exist.

**Para 2**

The simplest case of mathematical optimization is mathematical programming. Mathematical programming considers as its objective function a scalar real-valued function of real variables.

---

**Wikipedia – Mathematical Optimization (January, 2009)**

In mathematics, the simplest case of optimization, or mathematical programming, refers to the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This (a scalar real valued objective function) is actually a small subset of this field which comprises a large area of applied mathematics and generalizes to study of means to obtain "best available" values of some objective function given a defined domain where the elaboration is on the types of functions and the conditions and nature of the objects in the problem domain.
Given a function, \( \phi : S \rightarrow \mathbb{R} \), where \( \mathbb{R} \) is the real line, the optimization problem is to find \( x_0 \) such that \( x_0 \in S \) and \( \phi(x_0) \leq \phi(x) \) for every \( x \in S \) (minimization) or \( \phi(x_0) \geq \phi(x) \) for every \( x \in S \) (maximization).

\( S \) is usually a subset of \( \mathbb{R}^d \), the \( d \)-dimensional Euclidean space, which may be determined by a set of constraints.

The domain of \( \phi \), \( S \), is called the search space and the elements of \( S \) are called feasible solutions.

The function \( \phi \) in the operations research literature is called the objective function. In other literatures it is called an energy function (physics) or cost function (decision or game theory).

In the simplest cases, \( \phi \) is a twice-differentiable convex function over \( \mathbb{R} \) with no constraints on \( S \) in which case a minimum may be determined by setting the first partial derivatives of \( \phi \), \( \partial \phi / \partial x_i \), \( i = 1, \ldots, d \), equal to 0 and solving for \( x \) or \( \phi \) is a twice differentiable concave function over \( \mathbb{R} \) in which case a maximum may similarly be determined.

In general, the objective function will not be a convex or concave function, so that there may be a number of local maxima or minima.

For problems with twice-differentiable objective functions without constraints, the solutions can be found as stationary points of \( \phi \). The stationary points occur where the gradient of the objective function is zero, i.e., \( \phi = 0 \).

The Hessian matrix is used to classify each stationary point.

An optimization problem can be represented in the following way:

\[ \text{Given: a function } f : A \rightarrow \mathbb{R} \text{ from some set } A \text{ to the real numbers} \]

\[ \text{Sought: an element } x_0 \text{ in } A \text{ such that } f(x_0) \leq f(x) \text{ for all } x \text{ in } A \] ("minimization") or such that \( f(x_0) \geq f(x) \) for all \( x \) in \( A \) ("maximization").

Typically, \( A \) is some subset of the Euclidean space \( \mathbb{R}^d \), often specified by a set of constraints, equalities or inequalities that the members of \( A \) have to satisfy.

The domain \( A \) of \( f \) is called the search space, while the elements of \( A \) are called candidate solutions or feasible solutions.

The function \( f \) is called, variously, an objective function, cost function, energy function, or energy functional.

Generally, when the feasible region or the objective function of the problem does not present convexity, there may be several local minima and maxima.

For unconstrained problems with twice-differentiable functions, some critical points can be found by finding the points where the gradient of the objective function is zero (that is, the stationary points). …

Further, critical points can be classified using the definiteness of the Hessian matrix:...
[Hessian matrix omitted]

Para 4
If the Hessian is positive definite, the point is a local minimum. If the Hessian is negative definite, the point is a local maximum, and if the Hessian is indefinite, the point is a saddle point.

Para 5 (p.3-4)
Because the existence of derivatives may not always be assumed, many other methods depending on the smoothness of the objective function have been developed.

Some of the more well-known methods include:

- Conjugate gradient methods.
- Steepest descent methods.
- Interior point methods.
- Newton’s method.
- Quasi-Newton methods.
- Simplex methods.

[Hessian matrix (from Wikipedia – Hessian Matrix) omitted].

Wikipedia – Mathematical Optimization - Techniques

If the Hessian is positive definite at a critical point, then the point is a local minimum; if the Hessian matrix is negative definite, then the point is a local maximum; finally, if indefinite, then the point is some kind of saddle point.

However, existence of derivatives is not always assumed and many methods were devised for specific situations. The basic classes of methods, based on smoothness of the objective function, are:

[List of method classes omitted]

Actual methods falling somewhere among the categories above include:

- Bundle methods
- Conjugate gradient method
- Ellipsoid method
- Gradient descent aka steepest descent or steepest ascent
- Interior point methods
- Line search - a technique for one dimensional optimization, usually used as a subroutine for other, more general techniques.
- Nelder-Mead method aka the Amoeba method
- Newton’s method
- Quasi-Newton methods
- Simplex method
- Subgradient method - similar to gradient method in case there are no gradients

[Links to individual Wikipedia articles omitted]
Para 5 (cont.)

Constrained optimization problems can often be transformed into unconstrained problems with the help of Lagrange multipliers. This roadmap will briefly discuss some of these methods.

Wikipedia – Mathematical Optimization - Techniques

Should the objective function be convex over the region of interest, then any local minimum will also be a global minimum. There exist robust, fast numerical techniques for optimizing twice differentiable convex functions.

Constrained problems can often be transformed into unconstrained problems with the help of Lagrange multipliers.

Here are a few other popular methods:

- Ant colony optimization
- Beam search
- Bees algorithm
- Differential evolution
- Dynamic relaxation
- Evolution strategy
- Genetic algorithms
- Harmony search
- Hill climbing
- Particle swarm optimization
- Quantum annealing
- Simulated annealing
- Stochastic tunneling
- Tabu search

evolutionary algorithms, genetic algorithms, simulated annealing, and tabu search.
LAGRANGE MULTIPLIERS AND KARUSH–KUHN–TUCKER CONDITIONS
Para 1 (p. 4)
In the Introduction, simple multivariable calculus was the tool for optimizing a twice differentiable function. However, in nonlinear programming, optimizing that function may be subject to constraints. A method for doing this is called Lagrange multipliers.

Consider an objective function \( \phi(x) \) and a set of constraints \( g_k(x) = 0, k = 1, \ldots, m \). The domain of \( \phi \) is an open set that contains all points that satisfy the constraints. The functions \( \phi \) and \( g_k \) must have continuous first partial derivatives on the domain. Finally, the gradients of \( g_k \) must not be zero on the domain.

Then the Lagrangian, \( \Lambda \), is

\[
\Lambda(x, \lambda) = \phi(x) + \sum_{k=1}^{m} \lambda_k g_k(x).
\]

Here \( \lambda = (\lambda_1, \ldots, \lambda_m) \).

Wikipedia – Lagrange Multiplier
A more general formulation: The weak Lagrangian principle

Denote the objective function by \( f(x) \) and let the constraints be given by \( g_k(x) = 0 \). The domain of \( f \) should be an open set containing all points satisfying the constraints. Furthermore, \( f \) and the \( g_k \) must have continuous first partial derivatives and the gradients of the \( g_k \) must not be zero on the domain.

Now, define the Lagrangian, \( \Lambda \), as

\[
\Lambda(x, \lambda) = f + \sum_{k} \lambda_k g_k.
\]

\( k \) is an index for variables and functions associated with a particular constraint, \( k \).

\( \lambda \) without a subscript indicates the vector with elements \( \lambda_k \), which are taken to be independent variables.
Notice that $\frac{\partial \Lambda}{\partial x_i} = 0$ if and only if $\frac{\partial \Lambda}{\partial x_i} = - \sum_{k=1}^{m} \lambda_k \frac{\partial g_k}{\partial x_i}$ for $i = 1, \ldots, d$.

Furthermore, $\frac{\partial \Lambda}{\partial \lambda_k} = 0 \implies g_k = 0$.

Thus, these new normal equations, $\frac{\partial \Lambda}{\partial x_i} = 0$, $i = 1, \ldots, d$ and $\frac{\partial \Lambda}{\partial \lambda_k} = 0$, $k = 1, \ldots, m$, yield $d + m$ simultaneous equations.

The stationary points of this set of equations determine the optimization points of the constrained optimization problem. As before computing the Hessian will indicate whether the stationary point is maximum or minimum.

Observe that both the optimization criteria and constraints $g_k(x)$ are compactly encoded as stationary points of the Lagrangian:

$$\nabla_x \Lambda = 0 \quad \text{if and only if} \quad \nabla_x f = - \sum_k \lambda_k \nabla_x g_k,$$

$$\nabla_{\lambda} \Lambda = 0 \quad \text{implies} \quad g_k = 0.$$

Collectively, the stationary points of the Lagrangian, $\nabla \Lambda = 0$, give a number of unique equations totaling the length of $X$ plus the length of $\lambda$. 

\nabla_{\lambda} \Lambda = 0 \quad \text{implies} \quad g_k = 0.
Para 2
Consider now a generalization of the constrained optimization problem given above. Let \( \varphi(x) \) be the objective function. We wish to minimize \( \varphi(x) \) subject to the constraints \( g_k(x) \leq 0 \), \( k = 1, \ldots, m \) and \( h_j(x) = 0 \), \( j = 1, \ldots, l \).

\( g_k(x) \) are the inequality constraints and \( h_j(x) \) are the equality constraints.

William Karush first published the necessary conditions for this general equality–inequality constrained problem in his Masters thesis. However, it was not until Harold Kuhn and Albert Tucker rediscovered and popularized the necessary conditions in the Proceedings of the Second Berkeley Symposium that the power of the method was appreciated.

The necessary conditions are now known as the Karush–Kuhn–Tucker conditions.

Para 3
Let \( \varphi: \mathbb{R}^d \to \mathbb{R} \), \( g_k: \mathbb{R}^d \to \mathbb{R} \), \( k = 1, \ldots, m \), and \( h^j: \mathbb{R}^d \to \mathbb{R} \), \( j = 1, \ldots, l \). Assume these \( m + l + 1 \) functions are continuously differentiable at a point \( x^* \). If \( x^* \) is a local minimum that satisfies regularity conditions that will be specified shortly, then there exist constants \( \mu_k \), \( k = 1, \ldots, m \) and \( \lambda_j \), \( j = 1, \ldots, l \) such that

### Wikipedia – Karush-Kuhn-Tucker conditions (January 2009)
Let us consider the following nonlinear optimization problem:

Minimize \( f(x) \) subject to:

\[ g_i(x) \leq 0, \quad h_j(x) = 0 \]

where \( f(x) \) is the function to be minimized, \( g_i(x) \) \( (i = 1, \ldots, m) \) are the inequality constraints and \( h_j(x) \) \( (j = 1, \ldots, l) \) are the equality constraints, and \( m \) and \( l \) are the number of inequality and equality constraints, respectively.

The necessary conditions for this general equality-inequality constrained problem were first published in the Masters thesis of William Karush, although they only became renowned after a seminal conference paper by Harold W. Kuhn and Albert W. Tucker.

Suppose that the objective function, i.e., the function to be minimized, is \( f: \mathbb{R}^n \to \mathbb{R} \) and the constraint functions are \( g_i: \mathbb{R}^n \to \mathbb{R} \) and \( h^j: \mathbb{R}^n \to \mathbb{R} \). Further, suppose they are continuously differentiable at a point \( x^* \). If \( x^* \) is a local minimum that satisfies some regularity conditions, then there exist constants \( \mu_i \) \( (i = 1, \ldots, m) \) and \( \lambda_j \) \( (j = 1, \ldots, l) \) such that

Para 3 (cont.)

\[
\nabla \phi(x^*) + \sum_{b=1}^{m} \mu_b \nabla g_b(x^*) + \sum_{k=1}^{l} \lambda_k \nabla h_k(x^*) = 0
\]

and

\[ g_k(x^*) \leq 0, \text{ for all } k = 1, \ldots, m \]

\[ h_j(x^*) = 0, \text{ for all } j = 1, \ldots, l \]

and

\[ \mu_k \geq 0, \text{ for all } k = 1, \ldots, m \]

Stationarity condition

Primal feasibility

Dual feasibility

Complementary slackness

Para 4-8(p. 4)

The regularity conditions for a minimum point to satisfy the Karush–Kuhn–Tucker conditions are given below:

- **LICQ** - linear independence constraint qualification: the gradients of the active inequality constraints and the equality constraints are linearly independent at \( x^* \).

- **MFCQ** - Mangasarian–Fromowitz constraint qualification: the gradients of the active inequality constraints and the equality constraints are positive-linearly independent at \( x^* \).

- **CRCQ** - Constant rank constraint qualification: for each subset of the gradients of the active inequality constraints and the equality constraints, the rank at a vicinity of \( x^* \) is constant.

- **CPLD** - Constant positive linear dependence constraint qualification: for each subset of the gradients of the active inequality constraints and the equality constraints, if it is positive-linear dependent at \( x^* \), then it is positive-linear dependent at a vicinity of \( x^* \).

Stationarity

\[
\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x^*) = 0,
\]

Primal feasibility

**\( g_i(x^*) \leq 0, \text{ for all } i = 1, \ldots, m \)**

Dual feasibility

**\( h_j(x^*) = 0, \text{ for all } j = 1, \ldots, l \)**

Dual feasibility

\[ \mu_i \geq 0 \text{ (}\ i = 1, \ldots, m \text{)} \]

Complementary slackness

\[ \mu_i g_i(x^*) = 0 \text{ for all } i = 1, \ldots, m. \]

In order for a minimum point \( x^* \) to be KKT, it should satisfy some regularity condition, the most used ones are listed below:

- **Linear independence constraint qualification (LICQ):** the gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at \( x^* \).

- **Mangasarian-Fromowitz constraint qualification (MFCQ):** the gradients of the active inequality constraints and the gradients of the equality constraints are positive-linearly independent at \( x^* \).

- **Constant rank constraint qualification (CRCQ):** for each subset of the gradients of the active inequality constraints and the equality constraints, the rank at a vicinity of \( x^* \) is constant.

- **Constant positive linear dependence constraint qualification (CPLD):** for each subset of the gradients of the active inequality constraints and the equality constraints, if it is positive-linear dependent at \( x^* \), then it is positive-linear dependent at a vicinity of \( x^* \). (\( v_1, \ldots, v_n \) is positive-linear dependent if there exists \( a_1 \geq 0, \ldots, a_n \geq 0 \) not all zero such that \( a_1 v_1 + \ldots + a_n v_n = 0 \)).
Para 9-11

Slater condition - for a convex problem, there exists a point $x$ such that $h_i(x) = 0$ and $g_i(x) < 0$ for all constraints $g_i$ active in $x^*$.

Linearity constraints - If $\phi$ and $g_i$ are affine functions, then no other condition is needed to assure that the minimum point satisfies the Karush–Kuhn–Tucker conditions.

It is known that LICQ $\Rightarrow$ MFCQ $\Rightarrow$ CPLD and that LICQ $\Rightarrow$ CRCQ $\Rightarrow$ CPLD. The converses are not true. Also MFCQ is not equivalent to CRCQ.

The Karush–Kuhn–Tucker conditions are widely used in nonlinear programming. One prominent recent example is constrained nonlinear optimization used in the algorithm for determining the Support Vector Machine.

Wikipedia – Karush-Kuhn-Tucker conditions (cont.)

- Quasi-normality constraint qualification (QNCQ): if the gradientes of the active inequality constraints and the gradientes of the equality constraints are positive-linearly independent at $x^*$ with associated multipliers $\lambda_i$ for equalities and $\mu_j$ for inequalities than it doesn't exist a sequence $x_k \rightarrow x^*$ such that: $\lambda_i \neq 0 \Rightarrow \lambda_i h_i(x_k) > 0$ and $\mu_j \neq 0 \Rightarrow \mu_j g_j(x_k) > 0$.

- Slater condition: for a convex problem, there exists a point $x$ such that $h(x) = 0$ and $g_i(x) < 0$ for all $i$ active in $x^*$.

- Linearity constraints: If $f$ and $g$ are affine functions, then no other condition is needed to assure that the minimum point is KKT.

It can be shown that LICQ $\Rightarrow$ MFCQ $\Rightarrow$ CPLD $\Rightarrow$ QNCQ. LICQ $\Rightarrow$ CRCQ $\Rightarrow$ CPLD $\Rightarrow$ QNCQ (and the converses are not true), although MFCQ is not equivalent to CRCQ. …
Gradient Descent or Ascent
Para 3
Consider a function \( \varphi: \mathbb{R}^d \to \mathbb{R} \) that is defined and differentiable in a neighborhood of \( x^* \). Assume for purposes of discussion that \( x^* \) is the minimum of \( \varphi \). Suppose \( x_0 \) is a starting point in the neighborhood of \( x^* \). Then \( \varphi \) decreases from \( x_0 \) at the maximum rate in the direction of the negative gradient of \( \varphi(x_0) \).

This observation allows one to formulate the recursion:

\[
x_{n+1} = x_n - \gamma_n \nabla \varphi(x_n), \quad n \geq 0.
\]

\( \gamma_n \) is the step size of the recursion.

The gradient at \( x_n \) is orthogonal to the tangent plane of \( \varphi \) at \( x_n \). The step size \( \gamma_n \) is a decreasing sequence of real numbers that must be chosen carefully for convergence of \( x_n \) to \( x^* \).

Gradient ascent functions in a similar way with \( x_{n+1} = x_n + \gamma_n \nabla \varphi(x_n), \quad n \geq 0 \) where \( x^* \) is the maximum of \( \varphi \). This algorithm is also known as steepest descent or steepest ascent.

---

Wikipedia – Gradient Descent (2009)

Description

Gradient descent is based on the observation that if the real-valued function \( F(x) \) is defined and differentiable in a neighborhood of a point \( a \), then \( F(x) \) decreases fastest if one goes from \( a \) in the direction of the negative gradient of \( F \) at \( a \). It follows that, if

\[
b = a - \gamma \nabla F(a)
\]

for \( \gamma > 0 \) a small enough number, then \( F(a) > F(b) \). With this observation in mind, one starts with a guess \( x_0 \) for a local minimum of \( F \), and considers the sequence \( x_0, x_1, x_2, \ldots \) such that

\[
x_{n+1} = x_n - \gamma_n \nabla F(x_n), \quad n \geq 0.
\]

We have

\[
F(x_0) \geq F(x_1) \geq F(x_2) \geq \ldots,
\]

so hopefully the sequence \( (x_n) \) converges to the desired local minimum. Note that the value of the step size \( \gamma \) is allowed to change at every iteration.

\[\ldots\]

Note that the negative gradient at a point is orthogonal to the contour line going through that point.
The conjugate gradient method is a recursive numerical method for finding a solution to a system of linear equations whose matrix is symmetric and positive definite.

Consider the system of equations $Ax = b$ where $A$ is a $d \times d$ symmetric positive definite matrix. Thus $A^T = A$ and $x^T A x > 0$ for all nonzero vectors $x \in \mathbb{R}^d$.

Two positive vectors $u$ and $v$ are conjugate with respect to $A$ if $u^T A v = 0$.

Define the inner product of $u$ and $v$ with respect to $A$ as $\langle u, v \rangle_A := u^T A u = \langle A^T u, v \rangle = \langle u, A v \rangle$.

Since $A$ is symmetric and positive definite, the left-hand side defines an inner product.

So, two vectors are conjugate if they are orthogonal with respect to this inner product.

Suppose now that $\{u_k\}$ is a sequence of $d$ mutually conjugate vectors. Then $\{u_k\}$ spans $\mathbb{R}^d$ so that we may write the solution $x^*$ to the system $Ax = b$ as $x^* = \sum_{i=1}^d \lambda_i u_i$.

In this case we have $b = Ax^* = \sum_{i=1}^d \lambda_i A u_i$.

so that $u_k^T b = u_k^T A x^* = \sum_{i=1}^d \lambda_i u_k^T A u_i = \lambda_k u_k^T A u_k$.

Solving for $\lambda_k$, we have $\lambda_k = \frac{u_k^T b}{u_k^T A u_k} = \frac{\langle u_k, b \rangle_A}{\|u_k\|^2_A}$.
Now take 
\[ \varphi(x) = \frac{1}{2} x^T A x - b^T x, \ x \in \mathbb{R}^d. \]
This is a quadratic form for which \( x^* \) is the unique minimizer.

This suggests that we can use the gradient descent method to find \( x^* \).

We take the first basis vector to be \( u_1 = -b \).

If \( r_k = b - Ax_k \) is the residual at the \( k \)th iteration,

\( r_k \) is the negative gradient of \( \varphi \) at \( x_k \).

We take the direction closest to the gradient \( r_k \) that preserves the conjugacy constraint. Then

\[ u_{k+1} = r_k - \frac{u_k^T A r_k}{u_k^T A u_k} u_k. \]
The recursive equations are then:

$$
\lambda_k = \frac{r_k^T r_k}{u_k^T A u_k},
$$

$$
x_{k+1} = x_k - \lambda_k u_k,
$$

$$
r_{k+1} = r_k - \lambda_k A u_k,
$$

$$
\mu_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k},
$$

$$
u_{k+1} = r_{k+1} + \mu_k u_k.
$$

Then iterate this system of equations until $r_k$ is sufficiently small.

---

**Wikipedia - Conjugate Gradient Method (cont.)**

The resulting algorithm

After some simplifications, this results in the following algorithm for solving $Ax = b$ where $A$ is a real, symmetric, positive-definite matrix. The input vector $x_0$ can be an approximate initial solution or 0.

\[
\begin{align*}
r_0 &:= b - Ax_0 \\
p_0 &:= r_0 \\
k &:= 0 \\
&\text{repeat} \\
&\quad \text{if } r_{k+1} \text{ is "sufficiently small" then exit loop end if}
\end{align*}
\]

\[
\begin{align*}
\alpha_k &:= \frac{r_k^T r_k}{p_k^T A p_k} \\
x_{k+1} &:= x_k + \alpha_k p_k \\
r_{k+1} &:= r_k - \alpha_k A p_k \\
&\text{if } r_{k+1} \text{ is "sufficiently small" then exit loop end if}
\end{align*}
\]

\[
\begin{align*}
\beta_k &:= \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \\
p_{k+1} &:= r_{k+1} + \beta_k p_k \\
k &:= k + 1
\end{align*}
\]

\end{align*}
\]

end repeat

The result is $x_{k+1}$
LINEAR PROGRAMMING (p. 5-7)

Para 1
Linear programming is a technique for optimization of a linear objective function subject to linear equality and linear inequality constraints.

Given a polytope and a real-valued affine function on the polytope, a linear programming solution will be a point in the polytope where that function will have the smallest (or largest) value.

Suppose \( \phi(x) = c_1x_1 + c_2x_2 + \cdots + c_dx_d = c^T x \)
where vectors \( c \) and \( x \) are both column vectors. In the standard maximum problem, We wish to maximize \( c^T x \)
subject to some constraint of the form \( Ax \leq b \).
\( x \) represents the vector of variables to be determined, \( c \) and \( b \) are vectors of known coefficients, and \( A \) is a \( m \times d \) matrix of known coefficients.
In this case, \( c^T x \) is the objective function and \( Ax \leq b \) and \( x \geq 0 \) are the constraints that specify the convex polytope.

The standard minimum problem is to find \( y = (y_1, \ldots, y_d) \) to minimize \( b^T y \) subject to \( Ay \geq c \).

Wikipedia – Linear Programming

In mathematics, linear programming (LP) is a technique for optimization of a linear objective function, subject to linear equality and linear inequality constraints. Informally, linear programming determines the way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model and given some list of requirements represented as linear equations. More formally, given a polytope (for example, a polygon or a polyhedron), and a real-valued affine function

\[ f(x_1, x_2, \ldots, x_d) = c_1x_1 + c_2x_2 + \cdots + c_dx_d + d \]

defined on this polytope, a linear programming method will find a point in the polytope where this function has the smallest (or largest) value. Such points may not exist, but if they do, searching through the polytope vertices is guaranteed to find at least one of them.

Linear programs are problems that can be expressed in canonical form:

Maximize \( c^T x \)
Subject to \( Ax \leq b \).
\( x \) represents the vector of variables (to be determined), while \( c \) and \( b \) are vectors of (known) coefficients and \( A \) is a (known) matrix of coefficients. The expression to be maximized or minimized is called the objective function \( . c^T x \) in this case. The equations \( Ax \leq b \) are the constraints which specify a convex polyhedron over which the objective function is to be optimized.
LINEAR PROGRAMMING
Para 2

A vector \( x \) in the standard maximum problem or \( y \) in the standard minimum problem is said to be \emph{feasible} if it satisfies the corresponding constraints. The set of feasible vectors is the \emph{constraint set}. A linear programming problem is said to be \emph{feasible} if the constraint set is not empty; otherwise it is said to be \emph{infeasible}.  A feasible maximum (minimum) problem is said to be \emph{unbounded} if the objective function can assume arbitrarily large positive (negative) values at feasible vectors. If a problem is not unbounded it is said to be \emph{bounded}.

A minimum problem can be changed to a maximum problem by multiplying the objective function by \(-1\). Similarly, constraints of the form \( \sum_{j=1}^{d} a_{ij} x_j \geq b_i \) can be changed to the form \( \sum_{j=1}^{d} (-a_{ij}) x_j \leq -b_i \).

A variable \( x_j \) may not be restricted to be non-negative. In this case we can replace \( x_j \) by the difference of two variables \( u_j - v_j \) where both are restricted to be non-negative.

Thus corresponding to every maximum problem, called the \emph{primal problem}, there is a corresponding minimum problem which is said to be the \emph{dual problem}.

Ferguson – Linear Programming: A Concise Introduction (p. 8)

Terminology.

The function to be maximized or minimized is called the \textbf{objective function}.

A vector \( x \) for the standard maximum problem or \( y \) for the standard minimum problem, is said to be \emph{feasible} if it satisfies the corresponding constraints. The set of feasible vectors is \emph{the constraint set}. A linear programming problem is said to be \emph{feasible} if the constraint set is not empty; otherwise it is said to be \emph{infeasible}

A feasible maximum (resp. minimum) problem is said to be \emph{unbounded} if the objective function can assume arbitrarily large positive (resp. negative) values at feasible vectors; otherwise, it is said to be \emph{bounded} ...

All Linear Programming Problems Can Be Converted to Standard Form.

A minimum problem can be changed to a maximum problem by multiplying the objective function by \(-1\). Similarly, constraints of the form \( \sum_{i=1}^{n} a_{ij} x_i \geq b_i \) can be changed into the form \( \sum_{i=1}^{n} (-a_{ij}) x_i \leq -b_i \). Two other problems arise.

(1) \ldots

(2) \textbf{Some variable may not be restricted to be nonnegative}. An unrestricted variable, \( x_j \), may be replaced by the difference of two nonnegative variables, \( x_j = u_j - v_j \) where \( u_j \geq 0 \) and \( v_j \geq 0 \). ...

\textbf{Wikipedia – Linear Programming - Duality}

Every linear programming problem, referred to as a primal problem, can be converted into a dual problem, which provides an upper bound to the optimal value of the primal problem. …
LINEAR PROGRAMMING (cont.)
Simplex Algorithm (p. 6-7)
Para 4
The simplex algorithm is an algorithm to find a solution to a linear programming problem and is due to George Dantzig.

We consider the linear programming problem by maximizing \( c^T x \)
subject to \( Ax \leq b \) and \( x \geq 0 \).
Geometrically each inequality specifies a half-space in \( d \)-dimensional Euclidean space and their intersection is the set of all feasible values the variables can assume. The region is either empty, unbounded, or is a convex polytope. The set of points on which the objective function obtains the value \( v \) is defined by the hyperplane \( c^T x = v \). The solution to the linear programming problem will be by finding the largest \( v \) such that the hyperplane still intersects the feasible region. As \( v \) increases, the hyperplanes translate in the direction of the vector \( c \). The hyperplane corresponding to the largest \( v \) that still intersects the feasible region will intersect a vertex, a whole edge, or a face of the polytope. In the case of a edge or face, it is still the case that the endpoints of the edge or face will achieve the optimum value. Thus the optimum value of the objective function will always be achieved on one of the vertices of the polytope.

The simplex algorithm is based on rewriting the linear programming problem in an augmented form. The augmented form changes the basic inequalities to equalities by introducing the so-called slack variables.

In mathematical optimization theory, the simplex algorithm, created by the American mathematician George Dantzig in 1947, is a popular algorithm for numerical solution of the linear programming problem.

Overview
Consider a linear programming problem,

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \quad x \geq 0
\end{align*}
\]

In geometric terms, each inequality specifies a half-space in \( \mathbb{R}^n \)-dimensional Euclidean space, and their intersection is the set of all feasible values the variables can take. The region is either empty, unbounded, or a convex polytope. The set of points where the objective function obtains a given value \( v \) is defined by the hyperplane \( c^T x = v \). We are looking for the largest \( v \) such that the hyperplane still intersects the feasible region. As \( v \) increases, the hyperplanes translates in the direction of the vector \( c \). Intuitively, and indeed it can be shown by convexity, the last hyperplane to intersect the feasible region will either just graze a vertex of the polytope, or a whole edge or face. In the latter two cases, it is still the case that the endpoints of the edge or face will achieve the optimum value. Thus, the optimum value will always be achieved on one of the vertices of the polytope.

Wikipedia – Linear Programming – Augmented form
Linear programming problems must be converted into augmented form before being solved by the simplex algorithm. This form introduces slack variables to replace inequalities with equalities in the constraints.
LINEAR PROGRAMMING (cont.)
Simplex Algorithm (cont.)
Para 5

In matrix form the problem becomes

\[
\begin{bmatrix}
1 & -c^T & 0 \\
0 & A & I \\
0 & A & I
\end{bmatrix}
\begin{bmatrix}
x \\
x_s \\
x_s
\end{bmatrix}
\begin{bmatrix}
Z \\
x \\
x_s
\end{bmatrix}
= \begin{bmatrix}
0 \\
b \\
b
\end{bmatrix}
\]

where \(x\) are the variables from the standard form, \(x_s\) are the slack variables from the augmentation process, \(c\) contains the optimization coefficients, \(A\) and \(b\) describe the system of constraint equations, and \(Z\) is the variable to be optimized.

Suppose in the standard form of the problem there are \(d\) variables and \(m\) constraints, not counting the \(n\) non-negativity constraints. Generally, a vertex of the simplex corresponds to making \(d\) of the \(m + d\) total constraints tight, while adjacent vertices share \(d - 1\) tight constraints.

In the augmented form, this corresponds to setting \(m\) of the \(m + d\) variables (\(d\) original and \(m\) slack) to 0. Such a setting of the variables is called a basic solution. The \(m\) variables that are set to 0 are called the nonbasic variables. One can then solve the remaining \(d\) constraints, called the basic variables, that will be uniquely determined.

The simplex algorithm begins by finding a basic feasible solution. At each step, one basic and one nonbasic variable are chosen according to the pivot rule, and their roles are switched.

Wikipedia – Simplex Algorithm - Implementation

[One sentence omitted]

The problem can then be written as follows in matrix form:

\[
\begin{bmatrix}
1 & -c^T & 0 \\
0 & A & I \\
0 & A & I
\end{bmatrix}
\begin{bmatrix}
x \\
x_s \\
x_s
\end{bmatrix}
\begin{bmatrix}
Z \\
x \\
x_s
\end{bmatrix}
= \begin{bmatrix}
0 \\
b \\
b
\end{bmatrix}
\]

where \(x\) are the variables from the standard form, \(x_s\) are the introduced slack variables from the augmentation process, \(c\) contains the optimization coefficients, \(A\) and \(b\) describe the system of constraint equations, and \(Z\) is the variable to be maximized.

Suppose in the standard form of the problem there are \(n\) variables and \(m\) constraints, not counting the \(n\) nonnegativity constraints. Generally, a vertex of the simplex corresponds to making \(n\) of the \(m+n\) total constraints tight, while adjacent vertices share \(n-1\) tight constraints.

There is a little subtlety when such a point in \(n\) space does not fall in feasible region. Ignoring that, in the augmented form, this corresponds to setting \(m\) of the \(m+n\) variables (\(n\) original and \(m\) slack) to 0. We call such a setting of the variables a basic solution. The \(m\) variables which are purposely set to 0 are called the nonbasic variables. We can then solve for the remaining \(n\) constraints, called the basic variables, which will be uniquely determined, as we will be careful not to step out of the feasible region. The simplex algorithm begins by finding a basic feasible solution. At each step, one basic and one nonbasic variable are chosen according to the pivot rule, and their roles are switched.
LINEAR PROGRAMMING (cont.)
Simplex Algorithm (cont.)
Para 6

Klee and Minty\(^3\) developed a linear programming problem in which the polytope \(P\) is a distortion of a \(d\)-dimensional cube. In this case, the simplex method visits all \(2^d\) vertices before arriving at the optimal vertex. Thus the worst-case complexity for the simplex algorithm is exponential time.

However, the simplex method is remarkably efficient in practice. The simplex algorithm has polynomial-time average-case complexity under various distributions.

The computational formulation of the simplex algorithm will appear in another study.

LP - Interior Point Methods and Karmarkar’s Algorithm (p.7) Para 7

Karmarkar’s algorithm was introduced by Karmarkar\(^4\) as a polynomial-time algorithm for solving linear programming problems.

Although Dantzig’s simplex method usually performs well in practical problems, as noted above, it can in principle have exponential complexity.

Karmarkar’s algorithm guarantees polynomial time complexity and is reasonably efficient in practice.

If \(d\) is the number of variables and \(L\) is the number of bits input to the algorithm,

the runtime of Karmarkar’s algorithm is \(O(d^{3.5} L^2 \ln(L) \ln[\ln(L)])\).

Wikipedia – Simplex Algorithm – Overview (para. 5-6)

In 1972, Klee and Minty\(^2\) gave an example of a linear programming problem in which the polytope \(P\) is a distortion of an \(n\)-dimensional cube. They showed that the simplex method as formulated by Dantzig visits all \(2^n\) vertices before arriving at the optimal vertex. This shows that the worst-case complexity of the algorithm is exponential time.

Since then it has been shown that for almost every deterministic rule there is a family of simplices on which it performs badly. It is an open question if there is a pivot rule with polynomial time, or even sub-exponential worst-case complexity.

Nevertheless, the simplex method is remarkably efficient in practice. It has been known since the 1970s that it has polynomial-time average-case complexity under various distributions.

Wikipedia - Karmarkar's algorithm

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 \(O(n^6 L)\) such operations for the ellipsoid algorithm.

The runtime of Karmarkar's algorithm is thus \(O(n^{3.5} L^2 \ln L \ln \ln L)\) …
Karmarkar’s algorithm is an interior point method. That is, the current approximation to the solution does not follow the boundary of the feasible set as with the simplex method, but iterates through the interior of the feasible region and asymptotically approximates the solution.

In general, interior point methods (also called barrier methods) can be used to solve linear and nonlinear convex optimization problems.

A convex optimization problem can be transformed into a linear programming problem by minimizing or maximizing a linear function over a convex set. The idea of using interior methods was investigated in the early 1960s for general nonlinear programming problems, but were not pursued when more competitive methods were developed.

The introduction of Karmarkar’s algorithm established new interest in interior point methods.

Karmarkar’s algorithm is difficult to describe succinctly. A related interior point, but not polynomial time algorithm is the affine-scaling algorithm.

Pseudocode for the affine-scaling algorithm is given below.
Interior Point Methods and Karmarkar’s Algorithm  
Para 9

**Affine-Scaling Algorithm**

**Input:** $A$, $b$, $c$, stopping criterion, $\gamma$.

$k \leftarrow 0$

Do while stopping criterion is not satisfied

$$v^k \leftarrow b - Ax^k$$
$$D_v \leftarrow \text{diag}(v_1^k, \ldots, v_m^k)$$
$$h_x \leftarrow (A^T D_v^{-2} A)^{-1} c$$
$$h_v \leftarrow -Ah_x$$

If $h_x \geq 0$, then return unbounded

End If

$$\alpha \leftarrow \gamma \cdot \min\{-v^k_i / (h_v)_i | (h_v)_i < 0, i \in 1, \ldots, m\}$$

$$x^{k+1} \leftarrow x^k + \alpha h_x$$

$k \leftarrow k + 1$

End Do

DYNAMIC PROGRAMMING (p. 6) 
Para 1

Dynamic programming, pioneered by Richard Bellman in the 1940s Bellman, is a method of solving problems that exhibit the properties of overlapping subproblems and optimal substructure.

Although on the surface, dynamic programming would seem to be related to nonlinear and linear programming, it is actually an algorithm for speeding up computational efficiency. Generally the method is substantially more computationally efficient than naive methods.

Wikipedia – Karmarkar’s Algorithm – The Algorithm (cont.)

**Algorithm** Affine-Scaling

**Input:** $A$, $b$, $c$, $x^0$, stopping criterion, $\gamma$.

$k \leftarrow 0$

do while stopping criterion not satisfied

$$v^k \leftarrow b - Ax^k$$
$$D_v \leftarrow \text{diag}(v_1^k, \ldots, v_m^k)$$
$$h_x \leftarrow (A^T D_v^{-2} A)^{-1} c$$
$$h_v \leftarrow -Ah_x$$

if $h_v \geq 0$ then

return unbounded

end if

$$\alpha \leftarrow \gamma \cdot \min\{-v^k_i / (h_v)_i | (h_v)_i \leq 0, i = 1, \ldots, m\}$$

$$x^{k+1} \leftarrow x^k + \alpha h_x$$

$k \leftarrow k + 1$

end do

Wikipedia – Dynamic Programming

In mathematics and computer science, dynamic programming is a method of solving problems exhibiting the properties of overlapping subproblems and optimal substructure (described below) that takes much less time than naive methods. The term was originally used in the 1940s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another. …
**Dynamic Programming**

Para 1 (cont.)

Optimal substructure means that optimal solutions of subproblems can be used to find the optimal solutions of the overall problem.

Three steps are necessary:
1. Decompose the problem into smaller subproblems.
2. Optimally solve the subproblems using the three step process iteratively.
3. Use the optimal sub-solutions to build an optimal solution for the original problem.

Para 2

Overlapping subproblems means that the same subproblems are used to solve many different larger problems.

A computation wastes time if it is required to recompute optimal solutions to problems already computed.

In order to avoid this, optimal solutions to problems that have already been solved are saved. This process is known as memoization. If the problem needs to be solved later, one may simply look up the solution that has been saved.

Two approaches are common in dynamic programming: top-down and bottom-up. With a top-down approach, the problem is decomposed into subproblems, and these subproblems are solved and the solutions memoized, in case they need to be solved again.

With the bottom-up approach, all subproblems are solved in advance and then used to build up solutions to larger problems. It is not always very intuitive to recognize all the subproblems needed for solving the given problem.

---

**Wikipedia – Dynamic Programming**

*Optimal substructure* means that optimal solutions of subproblems can be used to find the optimal solutions of the overall problem. In general, we can solve a problem with optimal substructure using a three-step process:
1. Break the problem into smaller subproblems.
2. Solve these problems optimally using this three-step process recursively.
3. Use these optimal solutions to construct an optimal solution for the original problem.

The subproblems are, themselves, solved by dividing them into sub-subproblems, and so on, until we reach some simple case that is solvable in constant time.

To say that a problem has *overlapping subproblems* is to say that the same subproblems are used to solve many different larger problems. This applies whenever overlapping subproblems are present: a naive approach may waste time recomputing optimal solutions to subproblems it has already solved.

In order to avoid this, we instead save the solutions to problems we have already solved. Then, if we need to solve the same problem later, we can retrieve and reuse our already-computed solution. This approach is called memoization (not memorization, although this term also fits). …

Dynamic programming usually takes one of two approaches:
- **Top-down approach**: The problem is broken into subproblems, and these subproblems are solved and the solutions remembered, in case they need to be solved again. This is recursion and memoization combined together.
- **Bottom-up approach**: All subproblems that might be needed are solved in advance and then used to build up solutions to larger problems. This approach is slightly better in stack space and number of function calls, but it is sometimes not intuitive to figure out all the subproblems needed for solving the given problem.
CALCULUS OF VARIATIONS (p. 8)

Para 1
We only briefly mention calculus of variations. This area of mathematics is quite involved and this brief roadmap does not allow for a thorough treatment. Just as ordinary calculus deals with functions, calculus of variations deals with functionals. There are many possible functionals, but a common example would be integrals of an unknown function and/or its derivatives.

[Four apparently original sentences omitted.]

Para 2
Suppose $f$ is a real-valued, continuous, bounded function on an abstract topological space $\Omega$. Then the supremum norm (sup norm) $\|f\|_\infty = \sup\{f(\omega) : \omega \in \Omega\}$.

A functional $L(f)$ defined on a space of functions $\Xi$ with norm $\|\cdot\|_\Xi$ has a weak minimum at $f_0$ if there is a $\delta > 0$ such that if $\|f - f_0\|_\Xi < \delta$, then $L(f_0) \leq L(f)$. A weak maximum is defined similarly.

This abstract formulation may be replaced with a more typical formulation.

If $\Xi$ is the space of $r$ times continuously differentiable functions on a compact subset $E \subset \mathbb{R}$, then $\|f\|_\Xi = \sum_{k=0}^r \|f^{(k)}\|_\infty$.

A functional $L(f)$ has a strong minimum at $f_0$ if there is a $\delta > 0$ such that if $\|f - f_0\|_\infty < \delta$, then $L(f_0) \leq L(f)$.

Wikipedia – Calculus of Variations

Calculus of variations is a field of mathematics that deals with functionals, as opposed to ordinary calculus which deals with functions. Such functionals can for example be formed as integrals involving an unknown function and its derivatives. The interest is in extremal functions: those making the functional attain a maximum or minimum value.

Weak and strong extrema
Recall that the supremum norm for real, continuous, bounded functions on a topological space $X$ is defined as $\|y\|_\infty = \sup\{|y(x)| : x \in X\}$.

A functional $J(y)$ defined on some appropriate space of functions $V$ with norm $\|\cdot\|_V$ is said to have a weak minimum at the function $y_0$ if there exists some $\delta > 0$ such that for all functions $y$ with $\|y - y_0\|_V < \delta$, $J(y_0) \leq J(y)$.

Weak maxima are defined similarly, with the inequality in the last equation reversed.

In most problems, $V$ is the space of $r$-times continuously differentiable functions on a compact subset $E$ of the real line, with its norm given by $\|y\|_V = \sum_{n=0}^r \sup\{|y^{(n)}(x)| : x \in E\} = \sum_{n=0}^r \|y^{(n)}(x)\|_\infty$.

This norm is just the sum of the supremum norms of $y$ and its derivatives. A functional $J$ is said to have a strong minimum at $y_0$ if there exists some $\delta > 0$ such that for all functions $y$ with $\|y - y_0\|_\infty < \delta$, $J(y_0) \leq J(y)$. 
CALCULUS OF VARIATIONS (p. 8)

Para 3
A typical problem might be to find the stationary values of an integral of the form
\[ \mathcal{L}(f) = \int_a^b f(y, \dot{y}, x) \, dx \]
\[ \mathcal{L}(f) \] has an extremum only if the Euler–Lagrange differential equation is satisfied, i.e., if
\[ \frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{y}} \right) = 0. \]

The Euler–Lagrange equation plays a prominent role in classical mechanics and differential geometry.

Para 4
Suppose \( f \) is \( r \)-times continuously differentiable. The class of functions that are \( r \)-times differentiable on a compact set \( E \) is denoted by \( C^r(E) \).

Para 5
The fundamental lemma of the calculus of variations: Let \( f \in C^r [a, b] \) and let
\[ \int_a^b f(x) h(x) \, dx = 0 \]
for every \( h \in C^r [a, b] \) such that \( h(a) = h(b) = 0 \). Then
\[ f(x) = 0 \text{ on } (a, b); \]

Para 6
This lemma is used to prove that the extremal functions of
\[ \mathcal{L}(f) = \int_a^b f(y, \dot{y}, x) \, dx \]
are weak solutions of the Euler–Lagrange equation
\[ \frac{\partial f}{\partial y} - \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{y}} \right) = 0. \]

A generalization of the calculus of variations is called Morse Theory.

Wolfram MathWorld – Calculus of Variations

Mathematically, this involves finding stationary values of integrals of the form
\[ I = \int f(x, \dot{x}, x) \, dx \]
\( I \) has an extremum only if the Euler-Lagrange differential equation is satisfied, i.e., if
\[ \frac{d}{dx} \left( \frac{\partial f}{\partial \dot{x}} \right) - \frac{\partial f}{\partial x} = 0. \]

Wikipedia – Fundamental Lemma of the Calculus of Variations Statement

A function is said to be of class \( C^k \) if it is \( k \)-times continuously differentiable. …

Let \( f \) be of class \( C^k \) on the interval \([a, b]\). Assume furthermore that
\[ \int_a^b f(x) h(x) \, dx = 0 \]
for every function \( h \) that is \( C^k \) on \([a, b]\) with \( h(a) = h(b) = 0 \). Then the fundamental lemma of the calculus of variations states that \( f(x) \) is identically zero in the open interval \((a, b)\).

Wikipedia - Fundamental Lemma of the Calculus of Variations

This lemma is used to prove that extrema of the functional
\[ I[y] = \int_{x_0}^{x_f} L(t, y, \dot{y}) \, dt \]
are weak solutions of the Euler-Lagrange equation
\[ \frac{\partial L(t, y, \dot{y})}{\partial y} = \frac{d}{dt} \frac{\partial L(t, y, \dot{y})}{\partial \dot{y}}. \]

The Euler-Lagrange equation plays a prominent role in classical mechanics and differential geometry.

Wolfram MathWorld – Calculus of Variations

A generalization of calculus of variations known as Morse theory (and sometimes called "calculus of variations in the large") uses nonlinear techniques to address variational problems.
METAHEURISTIC ALGORITHMS (p. 8-10)
Evolutionary algorithms (p.8) – Omitted
Genetic Algorithms (p.8-9) - Omitted

Simulated Annealing (p. 9)
Annealing is defined as a process of heating then cooling for the purposes of softening or making a metal less brittle. The logical question that follows this definition is how this is related to optimization. The answer is that when a metal is heated, its atoms are freed from their initial position. In a process of slow cooling, the atoms settle into a structure of minimum energy. Thus, simulated annealing is an algorithm for finding a global minimum in a given system.

Simulated annealing is a probabilistic metaheuristic global optimization algorithm for locating a good approximation to the global minimum of a given function in a large search space.

For many problems, simulated annealing may be more effective than exhaustive enumeration provided that the goal is to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

During each step of the algorithm, the variable that will eventually represent the minimum is replaced by a random solution that is chosen according to a temperature parameter, $T$. As the temperature of the system decreases, the probability of higher temperature values replacing the minimum decreases, but it is always non-zero. The decrease in probability ensures a gradual decrease in the value of the minimum. However, the non-zero stipulation allows for a higher value to replace the minimum. Though this may sound like a flaw in the algorithm, it makes simulated annealing very useful because it allows for global minimums to be found rather than local ones. If during the course of the implementation of the algorithm a certain location (local minimum) has a lower temperature than its neighbors yet much higher than the overall lowest temperature (global minimum), this non-zero probability stipulation will allow for the value of the minimum to back track in a sense and become unstuck from local minima.

Tabu Search (p. 9-10) - omitted

Wikipedia – Simulated Annealing
Simulated annealing (SA) is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global minimum of a given function in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For certain problems, simulated annealing may be more effective than exhaustive enumeration — provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random “nearby” solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter $T$ (called the temperature), that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when $T$ is large, but increasingly "downhill" as $T$ goes to zero. The allowance for "uphill" moves saves the method from becoming stuck at local minima—which are the bane of greedier methods.
4. Analysis of Said and Wegman references and reading list

Said and Wegman used only seven references for their overview article, despite the expectation of 100 references set forth in the WIREs guidelines. All seven references appear to be derived directly from the unacknowledged antecedents, and thus arguably constitute false citations, insofar as the seven original sources may not have been directly consulted.

Each of the first four references, shown in the following table, show signs of simply being “carried over” from Wikipedia. The first three were actually cited within the strikingly similar antecedent passages, while the fourth (Karmarkar) is one of two main references for the corresponding Wikipedia article on Karmarkar’s Algorithm.

<table>
<thead>
<tr>
<th>Passage containing citation(s)</th>
<th>Sec</th>
<th>Pg</th>
<th>Reference(s)</th>
<th>Antecedent</th>
<th>SxS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klee and Minty developed a linear programming problem in which the polytope ( P ) is a distortion of a ( d )-dimensional cube.</td>
<td>4</td>
<td>6</td>
<td>3. Klee V, Minty GJ. How good is the simplex algorithm? In Shisha O, eds. Inequalities, III. NewYork: Academic Press; 1972, 159–175.</td>
<td>In 1972, Klee and Minty gave an example of a linear programming problem in which the polytope ( P ) is a distortion of an ( n )-dimensional cube. Ref: Greenberg cites: V. Klee and G.J. Minty. …</td>
<td>24</td>
</tr>
</tbody>
</table>
The last three references also appear to have been derived from unattributed antecedents. In the case of Bellman’s *Dynamic Programming*, the Wikipedia single reference seems to have been split into two, showing two different editions. The second (superfluous) reference has both the original and paperback publisher, with no information on the latter (as in Wikipedia). The repetition of “Bellman” in the running text may indicate an editing error following transcription.

Finally, the Tabu search section has no readily identifiable antecedent. However, the authors appear to have relied on incomplete and out-of-date sources to compile their summary. Fred Glover, and Fred Glover alone, is the universally acknowledged originator of the Tabu search concept. His work goes back to the 1970s and his seminal 1989 article was published nine years before the Glover-Lugana overview book. Moreover it is patently absurd to claim, after decades of successful application to a number of fields and problem areas, that this is a “recently developed method” whose “full implications and implementations … remain unknown”.

<table>
<thead>
<tr>
<th>Passage containing citation(s)</th>
<th>Sec</th>
<th>Pg</th>
<th>Reference(s)</th>
<th>Antecedent</th>
<th>SxS</th>
</tr>
</thead>
</table>

… [T]abu search works by examining neighboring configurations using its memory to avoid reaching a state of local optimality. Special override conditions are also included in case a more optimal state has been marked taboo. As such a recently developed method attributed to Glover and Laguna, the full implications and implementations of tabu search remain unknown.

| … [T]abu search works by examining neighboring configurations using its memory to avoid reaching a state of local optimality. Special override conditions are also included in case a more optimal state has been marked taboo. As such a recently developed method attributed to Glover and Laguna, the full implications and implementations of tabu search remain unknown. | 7   | 10 | 7. Glover F, Laguna M. Tabu Search. Norwell, MA: Kluwer; 1997. | It is worth noting that this “recently developed” meta-heuristic was described by Glover (alone) in 1989, 20 years ago and is the subject of literally 100s of papers. Relevant refs: F. Glover, Tabu search - part I (1989), ORSA Journal on Computing [Abstract and full reference] F. Glover, Tabu Search Fundamentals (1995). [Full PDF] | N/A |
In sharp contrast to the paltry reference list, Said and Wegman give an extensive list for “further reading”. This extensive bibliography also shows signs of having been derived from the unacknowledged antecedents analyzed in this report. Two example subject areas, linear programming and calculus of variations, were chosen at random for more detailed analysis. Works which appeared to be dedicated to these subject areas were checked against the relevant antecedents. As can be seen in the following table, all eight references can also be found listed in the reference lists for the corresponding online articles. This suggests that the reference lists of the antecedents may have been used to compile the bibliography for further reading.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Said and Wegman - Further Reading</th>
<th>Antecedent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Bliss GA. Calculus of Variations. Chicago, IL: Open Court; 1925.</td>
<td></td>
</tr>
</tbody>
</table>
References and further reading


Said and Wegman (2009) – Antecedents in order of appearance (with hyper links)

<table>
<thead>
<tr>
<th></th>
<th>Wikipedia – Mathematical Optimization (Jan. 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Wikipedia – Lagrange Multiplier (Jan. 2009)</td>
</tr>
<tr>
<td>4</td>
<td>Wikipedia – Gradient Descent (Nov. 2008)</td>
</tr>
<tr>
<td>5</td>
<td>Wikipedia – Conjugate Gradient Method (Jan. 2009)</td>
</tr>
<tr>
<td>6</td>
<td>Wikipedia – Linear Programming (Jan. 2009)</td>
</tr>
<tr>
<td>7</td>
<td>Tom Ferguson – Linear Programming: A Concise Introduction (Jan. 2009)</td>
</tr>
<tr>
<td>8</td>
<td>Wikipedia – Simplex Algorithm (Jan. 2009)</td>
</tr>
<tr>
<td>9</td>
<td>Wikipedia – Karmarkar’s Algorithm (Jan. 2009)</td>
</tr>
<tr>
<td>13</td>
<td>Wolfram MathWorld – Calculus of Variations</td>
</tr>
</tbody>
</table>

Further Reading

Discussion of the above analysis is given at:

Discussion and analysis of the WIREs Comp Stat article “Color Theory and Design” is given at:

An overview of scholarship issues in the Wegman report is found in:
- The SSWR Executive Summary is also available on its own as a separate document.

There is much more at DeepClimate.org, including:
- Wegman Report Update, part 2: GMU Dissertation Review
- Here is a list of related Deep Climate posts.

Acknowledgments

Thanks to John Mashey for a number of valuable suggestions and corrections, and to “amoeba” for bringing this to my attention all those months ago.