Improvements to Keyboard Optimization with Integer Programming

Andreas Karrenbauer\textsuperscript{1}  Antti Oulasvirta\textsuperscript{1,2}

\textsuperscript{1}Max Planck Center for Visual Computing and Communication, Saarbrücken, Germany

\textsuperscript{2}Aalto University, Helsinki, Finland

ACM UIST 2014
User Interface Optimization

Brute Force
Explicit enumeration
Nonpractical for UI (immense design spaces $>10^{26}$)
No guarantees when stopped prematurely

Random Search Heuristics
Random Enumeration
Practical
But by chance and no guarantees

A. Karrenbauer, A. Oulasvirta
User Interface Optimization

Brute Force

- Explicit enumeration
- Nonpractical for UI (immense design spaces $> 10^{26}$)
- No guarantees when stopped prematurely
User Interface Optimization

**Brute Force**
- Explicit enumeration
- Nonpractical for UI (immense design spaces $> 10^{26}$)
- No guarantees when stopped prematurely

**Random Search Heuristics**
- Random Enumeration
- Practical
- But by chance and no guarantees
User Interface Optimization

Brute Force
- Explicit enumeration
- Nonpractical for UI (immense design spaces $> 10^{26}$)
- No guarantees when stopped prematurely

Random Search Heuristics
- Random Enumeration
- Practical
- But by chance and no guarantees
Integer Programming

Given a set of constraints, decide whether an integer point satisfies all of them.

Benefits

1. Great modeling power
2. Efficient general purpose solvers
3. Exact Methods finding the global optimum in finite time
4. Rigorous bounds for the optimum even when interrupted prematurely
5. Complementary to other methods (e.g., Simulated Annealing)
6. Practical
Integer Programming

Given a set of constraints, decide whether an integer point satisfies all of them.

Benefits

1. Great modeling power
2. Efficient general purpose solvers

A. Karrenbauer, A. Oulasvirta
**Integer Programming**

Given a set of constraints, decide whether an integer point satisfies all of them.

**Benefits**

1. Great modeling power
2. Efficient general purpose solvers
3. Exact Methods finding the global optimum in finite time

![Graph showing objective vs. time with global optimum marker]
**Integer Programming**

Given a set of constraints, decide whether an integer point satisfies all of them.

**Benefits**

1. Great modeling power
2. Efficient general purpose solvers
3. Exact Methods finding the global optimum in finite time
4. Rigorous bounds for the optimum even when interrupted prematurely
Integer Programming

Given a set of constraints, decide whether an integer point satisfies all of them.

Benefits

1. Great modeling power
2. Efficient general purpose solvers
3. Exact Methods finding the global optimum in finite time
4. Rigorous bounds for the optimum even when interrupted prematurely
5. Complementary to other methods (e.g., Simulated Annealing)
**Integer Programming**

Given a set of constraints, decide whether an integer point satisfies all of them.

**Benefits**

1. Great modeling power
2. Efficient general purpose solvers
3. Exact Methods finding the global optimum in finite time
4. Rigorous bounds for the optimum even when interrupted prematurely
5. Complementary to other methods (e.g., Simulated Annealing)
6. Practical
Overview of the Approach

Steps

1. Definition of constraints and objective
2. Formalization with decision variables
3. Integer Programming Model
4. Refinement by reformulations and linearizations
5. IP-Solver
The Letter Assignment Problem – Constraints and Objective

Constraints: find an assignment (1-to-1 correspondence)

\[ \mathcal{L} = \{ \text{ABCDEFGHIJKL...} \} \]

Objective: maximize typing speed, i.e., minimize the expected inter-key intervals

\[ \sum_{k, \ell \in \mathcal{L}} p_{k, \ell} \cdot t_{f(k), f(\ell)} \]

\( p_{k, \ell} \) bigram probabilities
\( t_{i,j} \) inter-key intervals
The Letter Assignment Problem – Constraints and Objective

Constraints: find an assignment (1-to-1 correspondence)

\[ \mathcal{L} = \{ \text{ABCDEFGHI}, \text{JKLMNOPQR}, \text{STUVWX}, \text{YZ} \} \]

\[ f : \mathcal{L} \rightarrow \mathcal{K} \]

Objective: maximize typing speed, i.e., minimize the expected inter-key intervals

\[ \sum_{k,\ell \in \mathcal{L}} p_{k,\ell} \cdot t_{f(k),f(\ell)} \]

bigram probabilities \( p_{k,\ell} \)

inter-key intervals \( t_{i,j} \)
### The Letter Assignment Problem – Decision Variables

Model assignment by binary decision variables

\[
x_{\ell j} = \begin{cases} 
1 & \text{if letter } \ell \text{ is assigned to keyslot } j, \text{i.e., } f(\ell) = j \\
0 & \text{otherwise}
\end{cases}
\]
The Letter Assignment Problem – Decision Variables

Model assignment by binary decision variables

\[ x_{\ell j} = \begin{cases} 
1 & \text{if letter } \ell \text{ is assigned to keyslot } j, \text{i.e., } f(\ell) = j \\
0 & \text{otherwise} 
\end{cases} \]

Constraints

- Each letter \( \ell \) is assigned to exactly one keyslot
  \[ \forall \ell \in \mathcal{L} : \sum_{j \in \mathcal{K}} x_{\ell j} = 1 \]
- Each keyslot \( j \) contains exactly one letter
  \[ \forall j \in \mathcal{K} : \sum_{\ell \in \mathcal{L}} x_{\ell j} = 1 \]
The Letter Assignment Problem – Decision Variables

Model assignment by binary decision variables

\[ x_{\ell j} = \begin{cases} 
1 & \text{if letter } \ell \text{ is assigned to keyslot } j, \text{i.e., } f(\ell) = j \\
0 & \text{otherwise} 
\end{cases} \]

Constraints

- Each letter \( \ell \) is assigned to exactly one keyslot
  \[ \forall \ell \in \mathcal{L} : \sum_{j \in K} x_{\ell j} = 1 \]
- Each keyslot \( j \) contains exactly one letter
  \[ \forall j \in \mathcal{K} : \sum_{\ell \in \mathcal{L}} x_{\ell j} = 1 \]

Objective

expected inter-key interval

\[ \sum_{k,\ell \in \mathcal{L}} p_{k,\ell} \cdot t_{f(k),f(\ell)} \quad \sim \quad \sum_{i,j \in K} \sum_{k,\ell \in \mathcal{L}} p_{k,\ell} \cdot t_{i,j} \cdot x_{k,i} \cdot x_{\ell j} \]
The Letter Assignment Problem – IP Model

Quadratic Assignment Problem

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} p_{k\ell} t_{ij} x_{ki} x_{\ell j} \\
\text{s.t.} & \quad \sum_{\ell=1}^{n} x_{\ell j} = 1 & \forall j \in \{1, \ldots, n\} \\
& \quad \sum_{j=1}^{n} x_{\ell j} = 1 & \forall \ell \in \{1, \ldots, n\} \\
& \quad x_{\ell j} \in \{0, 1\} & \forall \ell, j \in \{1, \ldots, n\}
\end{align*}
\]
The Letter Assignment Problem – IP Model \rightarrow IP-Solver

Quadratic Assignment Problem

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} p_{k\ell} t_{ij} x_{ki} x_{\ell j}
\]

s.t. \[
\sum_{\ell=1}^{n} x_{\ell j} = 1 \quad \forall j \in \{1, \ldots, n\}
\]

\[
\sum_{j=1}^{n} x_{\ell j} = 1 \quad \forall \ell \in \{1, \ldots, n\}
\]

\[
x_{\ell j} \in \{0, 1\} \quad \forall \ell, j \in \{1, \ldots, n\}
\]
The Letter Assignment Problem – IP Model ↔ IP-Solver

Quadratic Assignment Problem

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} p_{k \ell} t_{ij} x_{ki} x_{\ell j}
\]

s.t. \[
\sum_{\ell=1}^{n} x_{\ell j} = 1 \quad \forall j \in \{1, \ldots, n\}
\]

\[
\sum_{j=1}^{n} x_{\ell j} = 1 \quad \forall \ell \in \{1, \ldots, n\}
\]

\[
x_{\ell j} \in \{0, 1\} \quad \forall \ell, j \in \{1, \ldots, n\}
\]
The Letter Assignment Problem – IP Models ↔ IP-Solver

Reformulations

\[
\begin{align*}
& \text{min} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} p_{k\ell} t_{ij} x_{ki} x_{\ell j} \\
& \text{s.t.} \sum_{\ell=1}^{n} x_{\ell j} = 1 \quad \forall j \in \{1, \ldots, n\} \\
& \sum_{j=1}^{n} x_{\ell j} = 1 \quad \forall \ell \in \{1, \ldots, n\} \\
& x_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, \ldots, n\}
\end{align*}
\]

\[
\begin{align*}
& \text{min} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} p_{k\ell} t_{ij} y_{ijk\ell} \\
& \text{s.t.} \sum_{\ell=1}^{n} x_{\ell j} = 1 \quad \forall j \in \{1, \ldots, n\} \\
& \sum_{j=1}^{n} x_{\ell j} = 1 \quad \forall \ell \in \{1, \ldots, n\} \\
& \sum_{k=1}^{n} y_{jk\ell} = x_{ij} \quad \forall i, j, \ell \in \{1, \ldots, n\} \\
& \sum_{\ell=1}^{n} y_{jk\ell} = x_{ij} \quad \forall j, k, \ell \in \{1, \ldots, n\} \\
& 0 \leq y_{jk\ell} \leq y_{jik\ell} \quad \forall i, j, k, \ell \in \{1, \ldots, n\}
\end{align*}
\]

Models can be solved in parallel.
Complementary to other approaches, such as random search heuristics.

A. Karrenbauer, A. Oulasvirta
The Letter Assignment Problem – IP Models ↔ IP-Solver

Reformulations

Models can be solved in parallel
Complementary to other approaches such as random search heuristics
Results

Virtual 3-row trapezoid
26 keys
Fitts’ model

Virtual hexagonal
29 keys
Fitts’ model

Physical 3-row standard
34 keys
Hiraga et al. 10-finger model
Results

Virtual 3-row trapezoid
26 keys
Fitts’ model

Virtual hexagonal
29 keys
Fitts’ model

Physical 3-row standard
34 keys
Hiraga et al. 10-finger model

Keyboards (wpm)
- qwerty (33.65)
- DSK (33.97)
- This (43.10)

- metropolis (42.04)
- justhci (42.38)
- This (42.65)

- qwerty (73.21)
- DSK (76.59)
- This (78.36)
Results

Virtual 3-row trapezoid
26 keys
Fitts’ model

Virtual hexagonal
29 keys
Fitts’ model

Physical 3-row standard
34 keys
Hiraga et al. 10-finger model

Keyboards (wpm)

- qwerty (33.65)
- DSK (33.97)
- This (43.10)
- any (< 44.82)

- metropolis (42.04)
- justhci (42.38)
- This (42.65)
- any (< 45.55)

- qwerty (73.21)
- DSK (76.59)
- This (78.36)
- any (< 79.95)
Summary

Take-home messages

- Integer Programming proved useful for UI Optimization
- IP provides rigorous mathematical guarantees for the optimum
- IP is complementary to other methods (e.g., random search heuristics)

http://resources.mpi-inf.mpg.de/keyboardoptimization/
(by Oct 15)
Summary

Take-home messages

- Integer Programming proved useful for UI Optimization
- IP provides rigorous mathematical guarantees for the optimum
- IP is complementary to other methods (e.g., random search heuristics)

Thank you for your attention!

http://resources.mpi-inf.mpg.de/keyboardoptimization/
(by Oct 15)