Fast, Large-Scale Model Predictive Control by Partial Enumeration

James B. Rawlings
Stephen J. Wright
Gabriele Pannocchia

COPTA Seminar Series
April 28, 2005
1. Overview
2. Full Enumeration Strategy
3. Small-scale Examples
4. Partial Enumeration Strategy
5. Large-scale Application
6. Conclusions
Separation of the control problem

Input/output description

State description

Regulation problem

Input/output description

State description

Regulation problem
Separation of the control problem

Input/output description

State description

input \( u \) output \( y \)

input \( u \) state \( x \) output \( y \)

\[ u \in \mathbb{R}^m \quad x \in \mathbb{R}^n \quad y \in \mathbb{R}^p \]
Separation of the control problem

Input/output description

State description

Regulation problem

Outline
Overview
Full Enumeration Strategy
Small-scale Examples
Partial Enumeration Strategy
Large-scale Application
Conclusions
Separation of the control problem

Estimation problem

- input: $u$
- process
- output: $y$
- estimator
- estimate: $\hat{x}$
Separation of the control problem

**Estimation problem**

```
input u → process → output y
```

- `estimator`
- `estimate \( \hat{x} \)`

**Control problem**

```
input u → regulator → process → estimator → output y
```

- `\( \hat{x} \)`
The Regulation Problem

1. Model and constraints
2. Objective function
3. Feedback
Linear dynamics and constraints

\( \frac{dx}{dt} = Ax + Bu \)
\( y = Cx \)
\( Du \leq d \)
\( Hx \leq h \)

\( x_{j+1} = Ax_j + Bu_j \)
\( y_j = Cx_j \)
\( Du_j \leq d \)
\( Hx_j \leq h \)
Nonlinear dynamics and constraints

\[
\frac{dx}{dt} = f(x, u) \\
y = g(x) \\
u \in \mathcal{U} \\
x \in \mathcal{X}
\]

\[
x_{j+1} = f(x_j, u_j) \\
y_j = g(x_j) \\
u_j \in \mathcal{U} \\
x_j \in \mathcal{X}
\]
Controller objective function

\[ \Phi(x, u(t)) = \sum_{k=0}^{\infty} L(x_k, u_k) \]

\[ L(x, u) = x' Q x + u' R u, \quad \text{quadratic measure common} \]
One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function.


Everything has been thought of before, but the problem is to think of it again.

— Goethe
One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement.
One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.
One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.

— Lee and Markus (1967)

*Foundations of Optimal Control Theory*
One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.


Everything has been thought of before, but the problem is to think of it again.

— Goethe
MPC by Enumeration

Overview
Full Enumeration Strategy
Small-scale Examples
Partial Enumeration Strategy
Large-scale Application
Conclusions

Feedback

Present
Future
Past

value of control objective

\[
\min_{u_k} \Phi(x, u)
\]

\[
x_{k+1} = Ax_k + Bu_k
\]

s.t.

\[
x_0 = \hat{x}_k
\]

\[
x_k \in \mathcal{X}
\]

\[
u_k \in \mathcal{U}
\]
Unexpected closed-loop behavior

- A finite horizon objective function may not give a stable controller!
Unexpected closed-loop behavior

- A finite horizon objective function may not give a stable controller!
- How is this possible?
Unexpected closed-loop behavior

- A finite horizon objective function may not give a stable controller!
- How is this possible?
Terminal constraint solution

- Adding a terminal constraint ensures stability
Adding a terminal constraint ensures stability
May cause infeasibility
Terminal constraint solution

- Adding a terminal constraint ensures stability
- May cause infeasibility
- Open-loop predictions not equal to closed-loop behavior

\[ \Phi_{k+2} \leq \Phi_{k+1} - L(x_{k+1}, u_{k+1}) \]

\[ \Phi_{k+1} \leq \Phi_k - L(x_k, u_k) \]
The infinite horizon ensures stability

\begin{align*}
\Phi_{k+2} &= \Phi_{k+1} - L(x_{k+1}, u_{k+1}) \\
\Phi_{k+1} &= \Phi_k - L(x_k, u_k)
\end{align*}
The infinite horizon ensures stability

Open-loop predictions equal to closed-loop behavior
The infinite horizon ensures stability
Open-loop predictions equal to closed-loop behavior
May be difficult to implement

\[ \Phi_{k+2} = \Phi_{k+1} - L(x_{k+1}, u_{k+1}) \]

\[ \Phi_{k+1} = \Phi_k - L(x_k, u_k) \]
Unconstrained solution: LQ regulator (Kalman, 1960)

\[ u = Kx \]

Constrained solution: MPC

\[ u_0 = K_i x + b_i \]

in which \( i \) enumerates different possible active sets for the inequality constraints [1]

There are \( 3^{mN} \) different active sets

\[
\begin{bmatrix}
\underline{u} \\
\underline{u} \\
\vdots \\
\underline{u}
\end{bmatrix} \land \begin{bmatrix}
\underline{u}^1 \\
\underline{u}^2 \\
\vdots \\
\underline{u}^m
\end{bmatrix} \land \begin{bmatrix}
\underline{u} \\
\underline{u} \\
\vdots \\
\underline{u}
\end{bmatrix}
\]
The active set table

\[ u_0 = K_i x + b_i \]
The active set table

\[ u_0 = K_i x + b_i \]

\[ N = 2 \]

<table>
<thead>
<tr>
<th>( i )</th>
<th>constraint set</th>
<th>( K_i )</th>
<th>( b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ ( \overline{u}, \overline{u} ) }</td>
<td>0</td>
<td>( \overline{u} )</td>
</tr>
<tr>
<td>2</td>
<td>{ ( \overline{u}, - ) }</td>
<td>0</td>
<td>( \overline{u} )</td>
</tr>
<tr>
<td>3</td>
<td>{ ( \overline{u}, u ) }</td>
<td>0</td>
<td>( \overline{u} )</td>
</tr>
<tr>
<td>4</td>
<td>{ - , ( \overline{u} ) }</td>
<td>( K_4 )</td>
<td>( b_4 )</td>
</tr>
<tr>
<td>5</td>
<td>{ - , - }</td>
<td>( K_5 )</td>
<td>( b_5 )</td>
</tr>
<tr>
<td>6</td>
<td>{ - , u }</td>
<td>( K_6 )</td>
<td>( b_6 )</td>
</tr>
<tr>
<td>7</td>
<td>{ u, ( \overline{u} ) }</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>8</td>
<td>{ u, - }</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>9</td>
<td>{ u, u }</td>
<td>0</td>
<td>( u )</td>
</tr>
</tbody>
</table>
The active set table

\[ u_0 = K_i x + b_i \]

For \( N = 2 \):

<table>
<thead>
<tr>
<th>( i )</th>
<th>constraint set</th>
<th>( K_i )</th>
<th>( b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( {\bar{u}, \bar{u}} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>2</td>
<td>( {\bar{u}, -} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>3</td>
<td>( {\bar{u}, u} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>4</td>
<td>( {-, \bar{u}} )</td>
<td>( K_4 )</td>
<td>( b_4 )</td>
</tr>
<tr>
<td>5</td>
<td>( {-, -} )</td>
<td>( K_5 )</td>
<td>( b_5 )</td>
</tr>
<tr>
<td>6</td>
<td>( {-, u} )</td>
<td>( K_6 )</td>
<td>( b_6 )</td>
</tr>
<tr>
<td>7</td>
<td>( {u, \bar{u}} )</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>8</td>
<td>( {u, -} )</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>9</td>
<td>( {u, u} )</td>
<td>0</td>
<td>( u )</td>
</tr>
</tbody>
</table>

For \( N = 4 \):

<table>
<thead>
<tr>
<th>( i )</th>
<th>constraint set</th>
<th>( K_i )</th>
<th>( b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( {\bar{u}, \bar{u}, \bar{u}, \bar{u}} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>2</td>
<td>( {\bar{u}, \bar{u}, \bar{u}, -} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>3</td>
<td>( {\bar{u}, \bar{u}, \bar{u}, u} )</td>
<td>0</td>
<td>( \bar{u} )</td>
</tr>
<tr>
<td>40</td>
<td>( {-, -, -, \bar{u}} )</td>
<td>( K_{40} )</td>
<td>( b_{40} )</td>
</tr>
<tr>
<td>41</td>
<td>( {-, -, -, -} )</td>
<td>( K_{41} )</td>
<td>( b_{41} )</td>
</tr>
<tr>
<td>42</td>
<td>( {-, -, -, u} )</td>
<td>( K_{42} )</td>
<td>( b_{42} )</td>
</tr>
<tr>
<td>79</td>
<td>( {u, u, u, \bar{u}} )</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>80</td>
<td>( {u, u, u, -} )</td>
<td>0</td>
<td>( u )</td>
</tr>
<tr>
<td>81</td>
<td>( {u, u, u, u} )</td>
<td>0</td>
<td>( u )</td>
</tr>
</tbody>
</table>
Example 1 — First order plus time delay

The first example is a first order plus time delay (FOPTD) system[2]

\[ G_1(s) = \frac{e^{-2s}}{10s + 1} \text{ sampled with } T_s = 0.25 \]

- The input is assumed to be constrained \(|u| \leq 1.5\)
- The control horizon is \(N = 4\)
Setpoint change and load disturbances

- In all simulations the setpoint is changed from 0 to 1 at time zero.
- At time 25 a load disturbance passing through the same dynamics as the plant of magnitude $-0.25$ enters the system.
- At time 50 the disturbance magnitude becomes $-1$ (which makes the setpoint 1 unreachable).
- Finally at time 75 the disturbance magnitude becomes $-0.25$ again.
FOPTD system: nominal case
MPC by Enumeration

Outline
Overview
Full Enumeration Strategy
Small-scale Examples
Partial Enumeration Strategy
Large-scale Application
Conclusions

FOPTD system: nominal case

---

**Controlled variable**

<table>
<thead>
<tr>
<th>Time</th>
<th>100</th>
<th>80</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setpoint</td>
<td>1.2</td>
<td>1</td>
<td>0.8</td>
<td>0.6</td>
<td>0.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

---

**Manipulated variable**

<table>
<thead>
<tr>
<th>Time</th>
<th>100</th>
<th>80</th>
<th>60</th>
<th>40</th>
<th>20</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setpoint</td>
<td>1.6</td>
<td>1.4</td>
<td>1.2</td>
<td>1</td>
<td>0.8</td>
<td>0.6</td>
</tr>
</tbody>
</table>

---

PID 2 PID 1 CLQ 2 CLQ 1
FOPTD system: noisy case

Outline
Overview
Full Enumeration Strategy
Small-scale Examples
Partial Enumeration Strategy
Large-scale Application
Conclusions
FOPTD system: noisy case
FOPTD system: effect of plant/model mismatch.
Under-damped system: nominal case.
Under-damped system: nominal case.
Under-damped system: noisy case.
Under-damped system: noisy case.
Under-damped system: effect of plant/model mismatch.

![Graph showing performance index vs relative mismatch for different control strategies](image)

- CLQ 1 (gain)
- PID 1 (gain)
- CLQ 1 (damping)
- PID 1 (damping)
The computational burden of CLQ is comparable to that of PID.

The CPU is a 1.7 GHz Athlon PC running Octave

<table>
<thead>
<tr>
<th></th>
<th>average CPU time (ms)</th>
<th>maximum CPU time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>0.05</td>
<td>0.10</td>
</tr>
<tr>
<td>CLQ</td>
<td>0.22</td>
<td>0.55</td>
</tr>
</tbody>
</table>
The challenge of large scale

- On-line QP solution may not be practical for large-scale applications

On-line QP solution may not be practical for large-scale applications because:

1. Computation time may be too long.
2. Full enumeration is not possible for typical large-sized applications.

Large-scale applications often have:

- $m = 5$, $N = 50$, $10^{11}$
- $m = 32$, $N = 15$, $10^{29}$
The challenge of large scale

- On-line QP solution may not be practical for large-scale applications
- Computation time may be too long
The challenge of large scale

- On-line QP solution may not be practical for large-scale applications
- Computation time may be too long
- Full enumeration is not possible!
On-line QP solution may not be practical for large-scale applications

- Computation time may be too long
- Full enumeration is not possible!

Typical large-sized applications:

\[ m = 5, \quad N = 50, \quad 10^{119} \]

\[ m = 32, \quad N = 15, \quad 10^{229} \]
The challenge of large scale

- On-line QP solution may not be practical for large-scale applications
- Computation time may be too long
- Full enumeration is not possible!
- Typical large-sized applications:
  \[ m = 5, \quad N = 50, \quad 10^{119} \]
  \[ m = 32, \quad N = 15, \quad 10^{229} \]

- Consider partial enumeration
Conjecture: In practice, the state might visit relatively few different active constraints! **SO...** don’t do the calculations for all possible active sets. Instead:

- Initialize by calculating and storing the table only for “likely” active sets;
- Given $\hat{x}$, search through the table and see if one of them is optimal;
- If so, use that control;
- If not, is there a feasible (suboptimal) solution? If so, use this one;
- If not, solve the MPC QP problem with small $N$ and add this solution to the table;
- Make room in the table by deleting the entry that was used least recently.
Choose a size for the table, and initialize its contents in a training period.

Add disturbances during training period to ensure that "useful" parts of the active-set space are sampled.

Larger disturbances tend to cause saturation of some inputs.

Saturation causes more active sets to be "interesting" because of small variations in the timepoint at which a constraint becomes active/inactive.
Searching and Managing the Table

- Maintain a counter of number of times an entry has been optimal
- Search the table in decreasing order of frequency of optimality
- No need to physically rearrange the table
- To make room for a new entry in the table, remove the least-recently-accessed active set (regardless of how frequently it has been accessed)
Computational Results: Problem 1

- Copolymerization reactor: 5 inputs, 18 states, 4 outputs, horizon $N = 50$.
- $10^{119}$ possible active sets.
Copolymerization reactor: 5 inputs, 18 states, 4 outputs, horizon $N = 50$.

$10^{119}$ possible active sets.

Training period: 28800 sample intervals (2 days, 1-minute intervals), 20 random disturbances applied. Visited a total of 376 active sets during this time.

Validation period: 43200 sample intervals (3 days), 30 random disturbances.

Considered tables of sizes 50, 100, and 200.
### MPC by Enumeration

#### Outline
- Overview
- Full Enumeration Strategy
- Small-scale Examples
- Partial Enumeration Strategy
- Large-scale Application
- Conclusions

---

## CPU time and suboptimality ratio of four solvers

<table>
<thead>
<tr>
<th></th>
<th>TAB50</th>
<th>TAB100</th>
<th>TAB200</th>
<th>qpsol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Max time (s)</strong></td>
<td>0.144</td>
<td>0.278</td>
<td>0.463</td>
<td>1.56</td>
</tr>
<tr>
<td><strong>Mean time (s)</strong></td>
<td>0.0075</td>
<td>0.0098</td>
<td>0.0129</td>
<td>1.02</td>
</tr>
<tr>
<td><strong>Active sets visited</strong></td>
<td>771</td>
<td>814</td>
<td>877</td>
<td>–</td>
</tr>
<tr>
<td><strong>Suboptimality</strong></td>
<td>1.0090</td>
<td>1.0088</td>
<td>1.0094</td>
<td>–</td>
</tr>
</tbody>
</table>
Cumulative distribution for the optimal table entry

Cumulative frequency vs. Number of entries scanned

- TAB200
- TAB100
- TAB50

Table entries:

<table>
<thead>
<tr>
<th>Number of entries scanned</th>
<th>TAB200</th>
<th>TAB100</th>
<th>TAB50</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.65</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>50</td>
<td>0.85</td>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>100</td>
<td>0.95</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>150</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Measuring table changeover

\[ N(k) \] the number of all table entries at time \( k \)

\[ D(k, j) \] the number of table entries at time \( k \) that are not in the table at time \( j \)

\[ R_d(k, j) \] relative table difference between times \( k \) and \( j \)

\[
R_d(k, 0) = \frac{D(k, 0)}{N(k)}
\]

\[
R_d(k, k - N) = \frac{D(k, k - N)}{N(k)}
\]
Disruptive Technology — Benefits of Enumeration

- Provides a single, scalable control technology ranging from the fastest SISO loop to the slowest, largest, MIMO dynamic plant optimization.
- Further work needed on several aspects:
  - Degeneracy
  - Backup strategy when no feasible entry in the table
- Requires lots more testing to fully characterize the reliability and performance.
Tom Badgwell, Aspentech

John Eaton for table coding advice

Financial support from TWMCC members and NSF grant #CTS-0105360