Some recent advances in project scheduling

Stefan Creemers
(June 27, 2018)
INTRODUCTION
Stefan who?

- PhD @ KU Leuven (2009)
Stefan who?

- PhD @ **KU Leuven** (2009)
- Visiting Professor @ **KU Leuven** (FT rank 94)
Stefan who?

- PhD @ KU Leuven (2009)
- Visiting Professor @ KU Leuven (FT rank 94)
- Full Professor @ IESEG (FT rank 61)
Stefan who?

- PhD @ KU Leuven (2009)
- Visiting Professor @ KU Leuven (FT rank 94)
- Full Professor @ IESEG (FT rank 61)
- Board of Directors PICS Belgium
Stefan who?

- PhD @ KU Leuven (2009)
- Visiting Professor @ KU Leuven (FT rank 94)
- Full Professor @ IESEG (FT rank 61)
- Board of Directors PICS Belgium
- Associate Editor INFORMS transactions on education
Stefan who?

- PhD @ KU Leuven (2009)
- Visiting Professor @ KU Leuven (FT rank 94)
- Full Professor @ IESEG (FT rank 61)
- Board of Directors PICS Belgium
- Associate Editor INFORMS transactions on education
- Research interests: project scheduling, logistics, queueing theory...
Stefan who?

- PhD @ KU Leuven (2009)
- Visiting Professor @ KU Leuven (FT rank 94)
- Full Professor @ IESEG (FT rank 61)
- Board of Directors PICS Belgium
- Associate Editor INFORMS transactions on education
- Research interests: project scheduling, logistics, queueing theory...
Some example projects

• Construction of the Rhein-Hellweg-Express
• Development of the Ebola vaccine
• Organizing the FIFA World Cup
Some example projects

• Construction of the Rhein-Hellweg-Express
Some example projects

• Construction of the Rhein-Hellweg-Express
• Development of the Ebola vaccine
Some example projects

- Construction of the Rhein-Hellweg-Express
- Development of the Ebola vaccine
- Organizing the FIFA World Cup

...
Project scheduling: important concepts
Project scheduling: important concepts

What?
Project scheduling: important concepts

What?

Activities
Project scheduling: important concepts

What?  Who?

Activities
Project scheduling: important concepts

What? Resources

Who? Activities
Project scheduling: important concepts

What?  Who?  When?

Activities  Resources
Project scheduling: important concepts

What?  Who?  When?

Activities  Resources  Schedule/Policy
Project scheduling: important concepts


Activities Resources Schedule/Policy
Project scheduling: important concepts


Activities Resources Schedule/Policy Makespan/NPV...
Project scheduling problems we’ll consider today

• Minimize makespan:
  – Deterministic activity durations:
    • No preemption: RCPSP
    • Preemption: PRCPSP
  – Stochastic activity durations:
    • No preemption: SRCPSP
    • Preemption: PSRCPSP

• Maximize NPV
  – Stochastic activity durations: SNPV

• All these problems are NP-hard!
Project scheduling problems we’ll consider today

• Minimize makespan:
Project scheduling problems we’ll consider today

- Minimize makespan:
  - Deterministic activity durations:
  - Stochastic activity durations:
  - All these problems are NP-hard!
Project scheduling problems we’ll consider today

• Minimize makespan:
  – Deterministic activity durations:
    • No preemption: RCPSP

• Maximize NPV
  – Stochastic activity durations:

• All these problems are NP-hard!
Project scheduling problems we’ll consider today

• Minimize makespan:
  – Deterministic activity durations:
    • No preemption: RCPSP
    • Preemption: PRCPSP

• Maximize NPV – Stochastic activity durations: SNPV

• All these problems are NP-hard!
Project scheduling problems we’ll consider today

- Minimize makespan:
  - Deterministic activity durations:
    - No preemption: **RCPSP**
    - Preemption: **PRCPSP**
  - Stochastic activity durations:
    - No preemption: **SRCPSP**
    - Preemption: **PSRCPSP**
Project scheduling problems we’ll consider today

• Minimize makespan:
  – Deterministic activity durations:
    • No preemption: RCPSP
    • Preemption: PRCPSP
  – Stochastic activity durations:
    • No preemption: SRCPSP
    • Preemption: PSRCSP

• Maximize NPV
  – Stochastic activity durations: SNPV
Project scheduling problems we’ll consider today

• Minimize makespan:
  – Deterministic activity durations:
    • No preemption: RCPSP
    • Preemption: PRCPSP
  – Stochastic activity durations:
    • No preemption: SRCPSP
    • Preemption: PSRCPS

• Maximize NPV
  – Stochastic activity durations: SNPV

• All these problems are NP-hard!
RCPSP

(Resource-Constrained Project Scheduling Problem)
RCPSP
(Resource-Constrained Project Scheduling Problem)
RCPSP
(Resource-Constrained Project Scheduling Problem)

Resource availability: 2
RCPSP
(Resource-Constrained Project Scheduling Problem)

Resource availability: 2

<table>
<thead>
<tr>
<th>ACT</th>
<th>DUR</th>
<th>RESOURCE USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Makespan

Resource availability

0 1 2 3 4 Makespan
PRCPSP

(Preemptive Resource-Constrained Project Scheduling Problem)
SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)
SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)

\[
\begin{array}{cccc}
\text{ACT} & \text{DUR} & \text{RESOURCE} & \text{USE} \\
1 & 4 & 1 & \\
2 & \{2,4\} & 1 & \\
3 & 2 & 2 & \\
4 & 2 & 1 & \\
\end{array}
\]

Resource availability: 2
SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)

Resource availability: 2

<table>
<thead>
<tr>
<th>ACT</th>
<th>DUR</th>
<th>RESOURCE USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>{2,4}</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)
PSRC PSP
(Preemptive Stochastic Resource-Constrained Project Scheduling Problem)

Resource availability: 2
SNPV

(Stochastic expected NPV maximization problem)
SNPV

(Stochastic expected NPV maximization problem)

Discount rate: 10%
Project payoff: 10
SNPV

(Stochastic expected NPV maximization problem)

<table>
<thead>
<tr>
<th>ACT</th>
<th>DUR</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>{2,4}</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-5</td>
</tr>
</tbody>
</table>

Discount rate: 10%
Project payoff: 10

NPV = 3.00
SNPV

(Stochastic expected NPV maximization problem)

Discount rate: 10%
Project payoff: 10

<table>
<thead>
<tr>
<th>ACT</th>
<th>DUR</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2,4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-5</td>
</tr>
</tbody>
</table>

NPV = 3.00
NPV = 2.71
SNPV

(Stochastic expected NPV maximization problem)

Discount rate: 10%
Project payoff: 10

ACT | DUR | COST
--- | --- | ---
1   | 4   | 0  
2   | {2,4} | 0  
3   | 1   | -5

eNPV = 2.86

NPV = 3.00
NPV = 2.71
THE RCPSP
The RCPSP: Facts & figures

• Google Scholar: 5370 hits
• Sciencedirect: 474 results
• Probably the most famous OR problem
• Solution heuristics implemented in software (even in Microsoft Project!)
• NP-hard! Easy to understand, hard to solve!
• Still 48 open problems for J60 (a set of benchmark problems)
The RCPSP: Facts & figures

- Google Scholar: 5370 hits
The RCPSP: Facts & figures

• **Google Scholar**: 5370 hits
• **Sciencedirect**: 474 results
The RCPSP: Facts & figures

- **Google Scholar**: 5370 hits
- **Sciencedirect**: 474 results
- Probably the most famous OR problem
The RCPSP: Facts & figures

- **Google Scholar**: 5370 hits
- **Sciencedirect**: 474 results
- Probably the most famous OR problem
- **Solution heuristics** implemented in software (even in Microsoft Project!)
The RCPSP: Facts & figures

• Google Scholar: 5370 hits
• Sciencedirect: 474 results
• Probably the most famous OR problem
• Solution heuristics implemented in software (even in Microsoft Project!)
• NP-hard! Easy to understand, hard to solve!
The RCPSP: Facts & figures

• **Google Scholar**: 5370 hits
• **Sciencedirect**: 474 results
• Probably the most famous OR problem
• **Solution heuristics** implemented in software (even in Microsoft Project!)
• **NP-hard**: Easy to understand, hard to solve!
• Still 48 open problems for **J60** (a set of benchmark problems)
The RCPSP: A brief (incomplete) timeline
The RCPSP: A brief (incomplete) timeline

Bowman (MIT): first optimal solution
The RCPSP: A brief (incomplete) timeline

1959
Bowman (MIT): first optimal solution

1983
Blazewicz (Poznan): proof that RCPSP is NP complete
The RCPSP: A brief (incomplete) timeline

- 1959: Bowman (MIT): first optimal solution
- 1983: Blazewicz (Poznan): proof that RCPSP is NP complete
- 1998: Bowman (MIT): first optimal solution
The RCPSP: A brief (incomplete) timeline

1959: Bowman (MIT): first optimal solution

1983: Blazewicz (Poznan): proof that RCPSP is NP complete

1992: Demeulemeester (KU Leuven): current state-of-the-art

1998: Age of heuristics

2000: Blazewicz (Poznan): proof that RCPSP is NP complete

Bowman (MIT): first optimal solution
The RCPSP: A brief (incomplete) timeline

- **1959**: Bowman (MIT): first optimal solution
- **1983**: Blazewicz (Poznan): proof that RCPSP is NP complete
- **1992**: Age of heuristics
- **1998**: Age of MILP solvers
- **2000**: Demeulemeester (KU Leuven): current state-of-the-art
- **2006**: Blazewicz (Poznan): proof that RCPSP is NP complete
- **Bowman (MIT): first optimal solution**
The RCPSP: A brief (incomplete) timeline

1959: Bowman (MIT): first optimal solution

1983: Blazewicz (Poznan): proof that RCPSP is NP complete

1992: Age of heuristics

1998: Age of MILP solvers

2000: Demeulemeester (KU Leuven): current state-of-the-art

2006: Creemers (IESEG): new state-of-the-art

2018: Age of heuristics
The RCPSP: new approach
The RCPSP: new approach

- Exact approach

Preliminary results:
- 17 times faster than current state-of-the-art
- Solutions to many unsolved benchmark problems

We expect final results to be even better
The RCPSP: new approach

• Exact approach
• Work in progress
The RCPSP: new approach

- Exact approach
- Work in progress
- Preliminary results:
  - 17 times faster than current state-of-the-art
  - Solutions to many unsolved benchmark problems
  - We expect final results to be even better
The RCPSP: new approach

- Exact approach
- Work in progress
- Preliminary results:
  - 17 times faster than current state-of-the-art
The RCPSP: new approach

- Exact approach
- Work in progress
- Preliminary results:
  - 17 times faster than current state-of-the-art
  - Solutions to many unsolved benchmark problems
The RCPSP: new approach

• Exact approach
• Work in progress
• Preliminary results:
  – 17 times faster than current state-of-the-art
  – Solutions to many unsolved benchmark problems
  – We expect final results to be even better
MARKOVIAN PERT NETWORKS: A NEW CTMC
Agenda

• CTMC of Kulkarni and Adlakha (1986)
• New CTMC
• Comparison of performance for the SRCPSP:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Comparison of performance for the SNPV:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Conclusion
Agenda

• **CTMC** of Kulkarni and Adlakha (1986)
• New CTMC
• Comparison of performance for the **SRCPSP**:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Comparison of performance for the **SNPV**:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Conclusion
Kulkarni & Adlakha (1986)


• First to study Markovian PERT networks

• Use of a CTMC to model a network

   There are up to 3^n states!

   Need for a strict partitioning of the statespace!
Kulkarni & Adlakha (1986)

Kulkarni & Adlakha (1986)

- 208 citations
Kulkarni & Adlakha (1986)


• 208 citations

• First to study Markovian PERT networks
Kulkarni & Adlakha (1986)

• 208 citations
• First to study Markovian PERT networks
• Use of a CTMC to model a network
Kulkarni & Adlakha (1986)

- 208 citations
- First to study Markovian PERT networks
- Use of a CTMC to model a network
- The states of the CTMC are defined by three sets: idle, ongoing, & finished activities
Kulkarni & Adlakha (1986)

- 208 citations
- First to study Markovian PERT networks
- Use of a CTMC to model a network
- The states of the CTMC are defined by three sets: idle, ongoing, & finished activities
  ⇒ For a project with $n$ activities there are up to $3^n$ states!
Example: State space

- An activity \( j \) is either:
  - Idle (\( q_j = 0 \))
  - Ongoing (\( q_j = 1 \))
  - Finished (\( q_j = 2 \))

- The state of the system is represented by a vector:
  \[ q = \{ q_1, q_2, \ldots, q_n \} \]

- Up to \( 3^n = 729 \) states

- Example feasible state:
  \[ q = \{ 2, 1, 1, 0, 0, 0 \} \]

- Example Infeasible state:
  \[ q = \{ 1, 1, 1, 1, 1, 1 \} \]
An activity $j$ is either:
- Idle ($\theta_j=0$)

Example state space:

- Example feasible state: $q = \{2,1,1,0,0,0\}$
- Example infeasible state: $q = \{1,1,1,1,1,1\}$
An activity $j$ is either:

- Idle ($\theta_j=0$)
- Ongoing ($\theta_j=1$)
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Ongoing ($\theta_j=1$)
  - Finished ($\theta_j=2$)

- The state of the system is represented by a vector: $q = \{q_1, q_2, \ldots, q_n\}$
- Up to $3^n = 729$ states
- Example feasible state: $q = \{2, 1, 1, 0, 0, 0\}$
- Example Infeasible state: $q = \{1, 1, 1, 1, 1, 1\}$
Example: State space

• An activity $j$ is either:
  – Idle ($\theta_j=0$)
  – Ongoing ($\theta_j=1$)
  – Finished ($\theta_j=2$)

• The state of the system is represented by a vector:
  $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$
Example: State space

An activity \( j \) is either:
- Idle (\( \theta_j=0 \))
- Ongoing (\( \theta_j=1 \))
- Finished (\( \theta_j=2 \))

The state of the system is represented by a vector:
\[ \theta = \{ \theta_1, \theta_2, \ldots, \theta_n \} \]

Up to \( 3^n = 729 \) states
An activity $j$ is either:
- Idle ($\theta_j=0$)
- Ongoing ($\theta_j=1$)
- Finished ($\theta_j=2$)

The state of the system is represented by a vector:
$\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$

Up to $3^n = 729$ states

Example feasible state:
$\theta = \{2,1,1,0,0,0\}$

Example infeasible state:
$\theta = \{1,1,1,1,1,1\}$
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Ongoing ($\theta_j=1$)
  - Finished ($\theta_j=2$)
- The state of the system is represented by a vector:
  $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$
- Up to $3^n = 729$ states
- Example feasible state: $\theta = \{2,1,1,0,0,0\}$
- Example Infeasible state: $\theta = \{0,0,0,2,2,2\}$
Agenda

• CTMC of Kulkarni and Adlakha (1986)
• New CTMC
• Comparison of performance for the SRCPSP:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Comparison of performance for the SNPV:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Conclusion
New CTMC

• We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986
New CTMC

• We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986
• In this new CTMC, states are defined by the set of finished activities
New CTMC

- We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986
- In this new CTMC, states are defined by the set of finished activities
  \[ \Rightarrow \text{up to } 2^n \text{ states (instead of } 3^n \text{ states) } \]
New CTMC

• We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986
• In this new CTMC, states are defined by the set of finished activities
  ⇒ up to $2^n$ states (instead of $3^n$ states)
  ⇒ Huge reduction in memory requirements (= THE bottleneck for CTMC of Kulkarni & Adlakha)
New CTMC

• We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986

• In this new CTMC, states are defined by the set of finished activities
  ⇒ up to $2^n$ states (instead of $3^n$ states)
  ⇒ Huge reduction in memory requirements (= THE bottleneck for CTMC of Kulkarni & Adlakha)

• A potential “drawback” is that the new CTMC allows activities to be preempted
Example: State space

- An activity $j$ is either:
  - Idle ($q_j = 0$)
  - Finished ($q_j = 1$)

- Up to $2^n = 64$ states

- Example feasible state: $q = \{1, 0, 0, 0, 0, 0\}$

- What activities are ongoing? 2? 3? 2 and 3?

- Preemption is possible
Example: State space

• An activity $j$ is either:
  – Idle ($\theta_j=0$)

---

Example feasible state: $q = \{1,0,0,0,0,0\}$

• What activities are ongoing? 2? 3? 2 and 3?
Example: State space

• An activity $j$ is either:
  – Idle ($\theta_j=0$)
  – Finished ($\theta_j=1$)

- Up to $2^n = 64$ states
- Example feasible state: $q = \{1,0,0,0,0,0\}$

- What activities are ongoing? 2? 3? 2 and 3?
- Preemption is possible
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Finished ($\theta_j=1$)
- Up to $2^n = 64$ states

Example feasible state: $q = \{1,0,0,0,0,0\}$

What activities are ongoing? 2? 3? 2 and 3?

Preemption is possible.
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Finished ($\theta_j=1$)
- Up to $2^n = 64$ states
- Example feasible state: \[ \theta = \{1,0,0,0,0,0,0\} \]
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Finished ($\theta_j=1$)
- Up to $2^n = 64$ states
- Example feasible state: $\theta = \{1,0,0,0,0,0,0\}$
- What activities are ongoing? 2? 3? 2 and 3?
Example: State space

- An activity $j$ is either:
  - Idle ($\theta_j=0$)
  - Finished ($\theta_j=1$)
- Up to $2^n = 64$ states
- Example feasible state: $\theta = \{1,0,0,0,0,0,0\}$
- What activities are ongoing? 2? 3? 2 and 3? 2
- Preemption is possible
Example: State space

In this state, it is optimal if activities 2 & 3 are ongoing
Example: State space

In this state, it is optimal if activities 2 & 3 are ongoing

Activity 2 finishes \(\Rightarrow\) we end up in state \(\theta = \{1,1,0,0,0,0\}\)
Example: State space

Activity 2 finishes \( \Rightarrow \) we end up in state \( \theta = \{1,1,0,0,0,0\} \)
Example: State space

Here, it is optimal if activity 4 is ongoing $\Rightarrow$ activity 3 is preempted!

Activity 2 finishes $\Rightarrow$ we end up in state $\theta = \{1,1,0,0,0,0,0\}$
Agenda

• CTMC of Kulkarni and Adlakha (1986)
• New CTMC

• Comparison of performance for the SRCPSP:
  – CPU times
  – Memory requirements
  – New state-of-the-art results

• Comparison of performance for the SNPV:
  – CPU times
  – Memory requirements
  – New state-of-the-art results

• Conclusion
Creemers (2015)

• Minimizing the expected makespan of a project with stochastic activity durations under resource constraints, *Journal of Scheduling*, 2015
Creemers (2015)

• Minimizing the expected makespan of a project with stochastic activity durations under resource constraints, *Journal of Scheduling*, 2015

• Current state-of-the-art for solving the SRCPSP
Creemers (2015)

- Minimizing the expected makespan of a project with stochastic activity durations under resource constraints, *Journal of Scheduling*, 2015
- Current state-of-the-art for solving the **SRCPSP**
- Uses CTMC of Kulkarni & Adlakha
Creemers (2015)

- Minimizing the expected makespan of a project with stochastic activity durations under resource constraints, *Journal of Scheduling*, 2015
- Current state-of-the-art for solving the SRCPSP
- Uses CTMC of Kulkarni & Adlakha
- Computational performance tested on well-known PSPLIB data sets (J30, J60, J90, & J120)
Creemers (2015)

- Minimizing the expected makespan of a project with stochastic activity durations under resource constraints, *Journal of Scheduling*, 2015
- Current state-of-the-art for solving the SRCPSP
- Uses CTMC of Kulkarni & Adlakha
- Computational performance tested on well-known PSPLIB data sets (J30, J60, J90, & J120)
- Bottleneck = memory requirements
## OLD CTMC

<table>
<thead>
<tr>
<th>Instances solved (out of 480)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>303</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>
### SRCPSP

#### 2015 (JOS) CPU Times

<table>
<thead>
<tr>
<th>Old CTMC</th>
<th>Instances solved (out of 480)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>303</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Old CTMC</th>
<th>Average CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>0.48</td>
</tr>
<tr>
<td>J60</td>
<td>1591</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>
## SRCPSP

### 2015 (JOS) VS new CTMC

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>OLD CTMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg CPU time (s) for same inst.</td>
<td>Average CPU time (s)</td>
</tr>
<tr>
<td>J30 0.02</td>
<td>J30 0.48</td>
</tr>
<tr>
<td>J60 81.6</td>
<td>J60 1591</td>
</tr>
<tr>
<td>J90 NA</td>
<td>J90 NA</td>
</tr>
<tr>
<td>J120 NA</td>
<td>J120 NA</td>
</tr>
</tbody>
</table>
### NEW CTMC

Avg CPU time (s) for same inst.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>0.02</td>
</tr>
<tr>
<td>J60</td>
<td>81.6</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>

### OLD CTMC

Average CPU time (s)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>0.48</td>
</tr>
<tr>
<td>J60</td>
<td>1591</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>

On average, we improve computation times by a factor of 19!
## Memory Requirements

### OLD CTMC

<table>
<thead>
<tr>
<th>Instances solved (out of 480)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>303</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>
### SRCPSP

**2015 (JOS) Memory Requirements**

<table>
<thead>
<tr>
<th>OLD CTMC</th>
<th>Instances solved (out of 480)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>303</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OLD CTMC</th>
<th>Average max # states (x1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>176</td>
</tr>
<tr>
<td>J60</td>
<td>374499</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>
**SRCPSP**

**2015 (JOS) VS new CTMC**

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>OLD CTMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg max # states (x1K) for inst.</td>
<td>Average max # states (x1000)</td>
</tr>
<tr>
<td>J30 1.99</td>
<td>J30 176</td>
</tr>
<tr>
<td>J60 508</td>
<td>J60 374499</td>
</tr>
<tr>
<td>J90 NA</td>
<td>J90 NA</td>
</tr>
<tr>
<td>J120 NA</td>
<td>J120 NA</td>
</tr>
</tbody>
</table>
SRCPSP
2015 (JOS) VS new CTMC

<table>
<thead>
<tr>
<th></th>
<th>NEW CTMC</th>
<th></th>
<th>OLD CTMC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg max # states (x1K) for = inst.</td>
<td>Average max # states (x1000)</td>
<td></td>
</tr>
<tr>
<td>J30</td>
<td>1.99</td>
<td>J30</td>
<td>176</td>
</tr>
<tr>
<td>J60</td>
<td>508</td>
<td>J60</td>
<td>374499</td>
</tr>
<tr>
<td>J90</td>
<td>NA</td>
<td>J90</td>
<td>NA</td>
</tr>
<tr>
<td>J120</td>
<td>NA</td>
<td>J120</td>
<td>NA</td>
</tr>
</tbody>
</table>

On average, we reduce memory requirements by a factor of 733!
### New CTMC Instances Solved

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>Instances solved (out of 480)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>480</td>
</tr>
<tr>
<td>J90</td>
<td>196</td>
</tr>
<tr>
<td>J120</td>
<td>10</td>
</tr>
</tbody>
</table>
SRCPSP

New CTMC Instances Solved

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>Instances solved (out of 480)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>480</td>
</tr>
<tr>
<td>J60</td>
<td>480</td>
</tr>
<tr>
<td>J90</td>
<td>196</td>
</tr>
<tr>
<td>J120</td>
<td>10</td>
</tr>
</tbody>
</table>

We are the first to solve instances of the J90 and J120 data sets to optimality!
Agenda

• CTMC of Kulkarni and Adlakha (1986)
• New CTMC
• Comparison of performance for the SRCPSP:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Comparison of performance for the SNPV:
  – CPU times
  – Memory requirements
  – New state-of-the-art results
• Conclusion
Current state-of-the-art for solving the SNPV

- Uses CTMC of Kulkarni & Adlakha
- Computational performance tested on dataset with different $n$ and Order Strength (OS)
- Bottleneck = memory requirements
Creemers, Leus, & Lambrecht (2010)

- Scheduling Markovian PERT networks to maximize the net present value, *Operations Research Letters*, 2010
Creemers, Leus, & Lambrecht (2010)

• Scheduling Markovian PERT networks to maximize the net present value, *Operations Research Letters*, 2010

• Current state-of-the-art for solving the SNPV
Creemers, Leus, & Lambrecht (2010)

- Scheduling Markovian PERT networks to maximize the net present value, *Operations Research Letters*, 2010
- Current state-of-the-art for solving the SNPV
- Uses CTMC of Kulkarni & Adlakha
Creemers, Leus, & Lambrecht (2010)

- Scheduling Markovian PERT networks to maximize the net present value, *Operations Research Letters*, 2010
- Current state-of-the-art for solving the SNPV
- Uses CTMC of Kulkarni & Adlakha
- Computational performance tested on dataset with different $n$ and Order Strength (OS)
Creemers, Leus, & Lambrecht (2010)

- Scheduling Markovian PERT networks to maximize the net present value, *Operations Research Letters*, 2010
- Current state-of-the-art for solving the SNPV
- Uses CTMC of Kulkarni & Adlakha
- Computational performance tested on dataset with different $n$ and Order Strength (OS)
- Bottleneck = memory requirements
# SNPV

## 2010 (ORL) Instances Solved

<table>
<thead>
<tr>
<th>OLD CTMC</th>
<th>Instances solved (out of 30)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS = 0.8</td>
</tr>
<tr>
<td>n = 10</td>
<td>30</td>
</tr>
<tr>
<td>n = 20</td>
<td>30</td>
</tr>
<tr>
<td>n = 30</td>
<td>30</td>
</tr>
<tr>
<td>n = 40</td>
<td>30</td>
</tr>
<tr>
<td>n = 50</td>
<td>30</td>
</tr>
<tr>
<td>n = 60</td>
<td>30</td>
</tr>
<tr>
<td>n = 70</td>
<td>30</td>
</tr>
</tbody>
</table>
### OLD CTMC

#### Instances solved (out of 30)

<table>
<thead>
<tr>
<th>n</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>30</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Average CPU time (s)

<table>
<thead>
<tr>
<th>n</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>7</td>
<td>2338</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>100</td>
<td>52268</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>2210</td>
<td>NA</td>
</tr>
<tr>
<td>70</td>
<td>3</td>
<td>17496</td>
<td>NA</td>
</tr>
</tbody>
</table>
# SNPV
## 2010 (ORL) VS new CTMC

### NEW CTMC

<table>
<thead>
<tr>
<th></th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 40</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>n = 50</td>
<td>0</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>n = 60</td>
<td>0</td>
<td>6</td>
<td>NA</td>
</tr>
<tr>
<td>n = 70</td>
<td>0</td>
<td>34</td>
<td>NA</td>
</tr>
</tbody>
</table>

### OLD CTMC

<table>
<thead>
<tr>
<th></th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>0</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>n = 40</td>
<td>0</td>
<td>7</td>
<td>2338</td>
</tr>
<tr>
<td>n = 50</td>
<td>0</td>
<td>100</td>
<td>52268</td>
</tr>
<tr>
<td>n = 60</td>
<td>1</td>
<td>2210</td>
<td>NA</td>
</tr>
<tr>
<td>n = 70</td>
<td>3</td>
<td>17496</td>
<td>NA</td>
</tr>
</tbody>
</table>
## SNPV

### 2010 (ORL) VS new CTMC

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>Old CTMC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average CPU time (s) for same instances</strong></td>
<td><strong>Average CPU time (s)</strong></td>
</tr>
<tr>
<td></td>
<td>OS = 0.8</td>
</tr>
<tr>
<td>n = 10</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>0</td>
</tr>
<tr>
<td>n = 40</td>
<td>0</td>
</tr>
<tr>
<td>n = 50</td>
<td>0</td>
</tr>
<tr>
<td>n = 60</td>
<td>0</td>
</tr>
<tr>
<td>n = 70</td>
<td>0</td>
</tr>
<tr>
<td>n = 10</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>0</td>
</tr>
<tr>
<td>n = 40</td>
<td>0</td>
</tr>
<tr>
<td>n = 50</td>
<td>0</td>
</tr>
<tr>
<td>n = 60</td>
<td>1</td>
</tr>
<tr>
<td>n = 70</td>
<td>3</td>
</tr>
</tbody>
</table>

On average, we improve computation times by a factor of 492!
### OLD CTMC

Instances solved (out of 30)

<table>
<thead>
<tr>
<th>n</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>30</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>
### SNPV 2010 (ORL) Memory Requirements

<table>
<thead>
<tr>
<th>OLD CTMC</th>
<th>Instances solved (out of 30)</th>
<th></th>
<th>Average max # states (x1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS = 0.8</td>
<td>OS = 0.6</td>
<td>OS = 0.4</td>
</tr>
<tr>
<td>n = 10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 20</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 40</td>
<td>30</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>n = 50</td>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>n = 60</td>
<td>30</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>n = 70</td>
<td>30</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>
## SNPV

### 2010 (ORL) VS new CTMC

<table>
<thead>
<tr>
<th>n</th>
<th>OLD CTMC</th>
<th>NEW CTMC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average max # states (x1000)</td>
<td>Avg max # states (x1000) for same inst.</td>
</tr>
<tr>
<td></td>
<td>OS = 0.8</td>
<td>OS = 0.6</td>
</tr>
<tr>
<td>n = 10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n = 40</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>n = 50</td>
<td>27</td>
<td>4346</td>
</tr>
<tr>
<td>n = 60</td>
<td>287</td>
<td>216028</td>
</tr>
<tr>
<td>n = 70</td>
<td>287</td>
<td>216028</td>
</tr>
</tbody>
</table>
On average, we reduce memory requirements by a factor of 403!
## NEW CTMC

<table>
<thead>
<tr>
<th>Instances solved (out of 30)</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 20</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 40</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 50</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 60</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 70</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>
### SNPV

**New CTMC CPU Times**

**NEW CTMC**

<table>
<thead>
<tr>
<th>Instances solved (out of 30)</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 20</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 40</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 50</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 60</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 70</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

**NEW CTMC**

<table>
<thead>
<tr>
<th>Average CPU time (s)</th>
<th>OS = 0.8</th>
<th>OS = 0.6</th>
<th>OS = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 40</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>n = 50</td>
<td>0</td>
<td>1</td>
<td>476</td>
</tr>
<tr>
<td>n = 60</td>
<td>0</td>
<td>11</td>
<td>16869</td>
</tr>
<tr>
<td>n = 70</td>
<td>0</td>
<td>99</td>
<td>263012</td>
</tr>
</tbody>
</table>
SNPV

New CTMC CPU Times

<table>
<thead>
<tr>
<th>NEW CTMC</th>
<th>Instances solved (out of 30)</th>
<th>Average CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS = 0.8</td>
<td>OS = 0.6</td>
</tr>
<tr>
<td>n = 10</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 20</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 40</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 50</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 60</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>n = 70</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

CPU times have become the new bottleneck
SNPV

To preempt or not to preempt?

• If an activity has a zero cost, it is optimal to start that activity as early as possible.
• If at time $t$, activity $i$ is preempted, the remainder of activity $i$ joins the set of eligible activities.
• The remainder of activity $i$ has a zero cost (the cost has already been incurred at the start of activity $i$).

$\Rightarrow$ It is optimal to start the remainder of activity $i$ at time $t$.

$\Rightarrow$ It is optimal not to preempt activity $i$. 
SNPV

To preempt or not to preempt?

• If an activity has a zero cost, it is optimal to start that activity as early as possible
SNPV

To preempt or not to preempt?

• If an activity has a zero cost, it is optimal to start that activity as early as possible.
• If at time $t$ activity $i$ is preempted, the remainder of activity $i$ joins the set of eligible activities.
SNPV

To preempt or not to preempt?

• If an activity has a zero cost, it is optimal to start that activity as early as possible
• If at time $t$ activity $i$ is preempted, the remainder of activity $i$ joins the set of eligible activities
• The remainder of activity $i$ has a zero cost (the cost has already been incurred at the start of activity $i$)
To preempt or not to preempt?

• If an activity has a **zero cost**, it is **optimal** to start that activity as early as possible.
• If at time \( t \) activity \( i \) is preempted, the remainder of activity \( i \) joins the set of eligible activities.
• The remainder of activity \( i \) has a **zero cost** (the cost has already been incurred at the start of activity \( i \)).

\[ \Rightarrow \text{It is optimal to start the remainder of activity } i \text{ at time } t \]
To preempt or not to preempt?

- If an activity has a zero cost, it is optimal to start that activity as early as possible.
- If at time $t$ activity $i$ is preempted, the remainder of activity $i$ joins the set of eligible activities.
- The remainder of activity $i$ has a zero cost (the cost has already been incurred at the start of activity $i$).

⇒ It is optimal to start the remainder of activity $i$ at time $t$.
⇒ It is optimal not to preempt activity $i$. 
Agenda

- CTMC of Kulkarni and Adlakha (1986)
- New CTMC
- Comparison of performance for the SRCPSP:
  - CPU times
  - Memory requirements
  - New state-of-the-art results
- Comparison of performance for the SNPV:
  - CPU times
  - Memory requirements
  - New state-of-the-art results
- Conclusion
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
• New state-of-the-art for solving the SRCPSP and the SNPV
• Bottleneck shifted from memory requirements to CPU times
• Only "drawback" is that the new CTMC allows activities to be preempted
• We prove that there is no preemption when solving the SNPV
Conclusion

- New CTMC that only keeps track of finished activities
- Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
- New state-of-the-art for solving the SRCPSP and the SNPV
- Bottleneck shifted from memory requirements to CPU times
- Only “drawback” is that the new CTMC allows activities to be preempted
- We prove that there is no preemption when solving the SNPV
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
• New state-of-the-art for solving the SRCPSP and the SNPV
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
• New state-of-the-art for solving the SRCPSP and the SNPV
• Bottleneck shifted from memory requirements to CPU times
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
• New state-of-the-art for solving the SRCPSP and the SNPV
• Bottleneck shifted from memory requirements to CPU times
• Only “drawback” is that the new CTMC allows activities to be preempted
Conclusion

• New CTMC that only keeps track of finished activities
• Significantly reduces memory requirements when compared with CTMC of Kulkarni & Adlakha
• New state-of-the-art for solving the SRCPSP and the SNPV
• Bottleneck shifted from memory requirements to CPU times
• Only “drawback” is that the new CTMC allows activities to be preempted
• We prove that there is no preemption when solving the SNPV
MOMENTS & DISTRIBUTION OF PROJECT NPV
Agenda

• Introduction
• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages
• General projects
• Conclusions
Agenda

• Introduction

• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages

• General projects

• Conclusions
Introduction
Introduction

• We study the NPV of a project where:
  – Activities have general duration distributions
  – Cash flows are incurred during the lifetime of the project
Introduction

• We study the NPV of a project where:
  – Activities have general duration distributions
  – Cash flows are incurred during the lifetime of the project

• For such settings, most of the literature has focused on determining the expected NPV (eNPV) of a project
Introduction

• We study the NPV of a project where:
  – Activities have general duration distributions
  – Cash flows are incurred during the lifetime of the project

• For such settings, most of the literature has focused on determining the expected NPV (eNPV) of a project

• Higher moments/distribution of project NPV are currently determined using Monte Carlo simulation
Introduction

• We study the NPV of a project where:
  – Activities have general duration distributions
  – Cash flows are incurred during the lifetime of the project
• For such settings, most of the literature has focused on determining the expected NPV (eNPV) of a project
• Higher moments/distribution of project NPV are currently determined using Monte Carlo simulation
• We develop exact, closed-form expressions for the moments of project NPV & develop an accurate approximation of the NPV distribution itself
Agenda

• Introduction
• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages
• General projects
• Conclusions
NPV of a single cash flow obtained after a single stage
NPV of a single cash flow obtained after a single stage

- $c_w =$ cash flow incurred at start of stage $w$

\[
\begin{align*}
\text{stage } \ W & \\
c_w & \\
\end{align*}
\]
NPV of a single cash flow obtained after a single stage

- $c_w = \text{cash flow incurred at start of stage } w$
- $\nu_w = \text{NPV of cash flow } c_w$
NPV of a single cash flow obtained after a single stage

- $c_w$ = cash flow incurred at start of stage $w$
- $\nu_w$ = NPV of cash flow $c_w$
- $f_w(t)$ = distribution of time until cash flow $c_w$ is incurred
NPV of a single cash flow obtained after a single stage

\[ \nu_w = c_w \int_0^\infty f_w(t) e^{-rt} \, dt \]

- \( c_w \) = cash flow incurred at start of stage \( w \)
- \( \nu_w \) = NPV of cash flow \( c_w \)
- \( f_w(t) \) = distribution of time until cash flow \( c_w \) is incurred
- \( r \) = discount rate
NPV of a single cash flow obtained after a single stage

- \( c_w \) = cash flow incurred at start of stage \( w \)
- \( \nu_w \) = NPV of cash flow \( c_w \)
- \( f_w(t) \) = distribution of time until cash flow \( c_w \) is incurred
- \( r \) = discount rate
- \( M_{f_w(t)}(-r) \) = moment generating function of \( f_w(t) \) about \(-r\)

\[
\nu_w = c_w \int_0^\infty f_w(t) e^{-rt} \, dt
\]

\[
\nu_w = c_w M_{f_w(t)}(-r)
\]
NPV of a single cash flow obtained after a single stage

\[ \nu_w = c_w \int_0^\infty f_w(t) e^{-rt} \, dt \]

\[ \nu_w = c_w M_{f_w(t)}(-r) \]

\[ \nu_w = c_w \phi_w(r) \]

- \( c_w \) = cash flow incurred at start of stage \( w \)
- \( \nu_w \) = NPV of cash flow \( c_w \)
- \( f_w(t) \) = distribution of time until cash flow \( c_w \) is incurred
- \( r \) = discount rate
- \( M_{f_w(t)}(-r) \) = moment generating function of \( f_w(t) \) about \(-r\)
- \( \phi_w(r) \) = discount factor for stage \( w \)
NPV of a single cash flow obtained after a single stage

\[ v_w = c_w \phi_w(r) \]
NPV of a single cash flow obtained after a single stage

\[ v_w = c_w \phi_w(r) \]

Using discount factor \( \phi_w(r) \), we can obtain the moments of the NPV:

- \( \mu_w = c_w \phi_w(r) \)
- \( \sigma_w^2 = c_w^2 (\phi_w(2r) - \phi_w^2(r)) \)
- \( \gamma_w = c_w^3 (\phi_w(3r) - 3\phi_w(2r)\phi_w(r) + 2\phi_w^3(r)) \sigma_w^{-3} \)
- \( \theta_w = c_w^4 (\phi_w(4r) - 4\phi_w(3r)\phi_w(r) + 6\phi_w(2r)\phi_w^2(r) - 3\phi_w^4(r)) \sigma_w^{-4} \)
Now
\[ v_w = c_w \phi_w(r) \]

Using discount factor \( \phi_w(r) \), we can obtain the moments of the NPV:

- \( \mu_w = c_w \phi_w(r) \)
- \( \sigma^2_w = c^2_w (\phi_w(2r) - \phi^2_w(r)) \)
- \( \gamma_w = c^3_w (\phi_w(3r) - 3\phi_w(2r)\phi_w(r) + 2\phi^3_w(r)) \sigma^{-3}_w \)
- \( \theta_w = c^4_w (\phi_w(4r) - 4\phi_w(3r)\phi_w(r) + 6\phi_w(2r)\phi^2_w(r) - 3\phi^4_w(r)) \sigma^{-4}_w \)

The CDF & PDF of the NPV of \( c_w \) are:

- \( G_w(v) = 1 - F_w \left( \ln \left( \frac{c_w}{v} \right) r^{-1} \right) \)
- \( g_w(v) = \frac{f_w(\ln(\frac{c_w}{v})r^{-1})}{|r|_v} \)
Agenda

• Introduction

• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages

• General projects

• Conclusions
NPV of a single cash flow obtained after multiple stages
NPV of a single cash flow obtained after multiple stages

\[
v_w = c_w \sum \phi_i(r)\]

\[
v_w = c_w \phi_1, w(r)
\]

We can obtain the moments of the NPV of cash flow \(c_w\):

\[
\mu_w = c_w \phi_1, w(r)
\]

\[
\sigma^2_w = c_w^2 (\phi_1, 2w - \phi_1, w)
\]

\[
c_w
\]
NPV of a single cash flow obtained after multiple stages

\[ \nu_w \rightarrow \text{stage 1} \rightarrow \ldots \rightarrow \text{stage } w - 1 \rightarrow \text{stage } w \]

\[ \nu_w c_w \phi_1(r) \phi_{w-1}(r) \phi_w(r) \]

now

\[ \mu_w = \nu_w \phi_1(r) \]

\[ \sigma_w^2 = \nu_w^2 (\phi_1(r) - \phi_w(r)) \]

\[ \ldots \]
NPV of a single cash flow obtained after multiple stages

\[ v_w = c_w \phi_1(r) \cdots \phi_w(r) \]

- We can obtain the moments of the NPV of cash flow \( c_w \):
  - \( \mu_w = c_w \phi_1, w(r) \)
  - \( \sigma_w^2 = c_w^2 (\phi_1, w^2 r - \phi_1, w r) \)
  - …
NPV of a single cash flow obtained after multiple stages

Now $f_1(t)$ stage 1 $\cdots$ stage $w-1$ $f_w(t)$ stage $w$

$v_w \phi_1(r)$ $\phi_{\ldots}(r)$ $\phi_{w}(r)$

$\mu_w = c_w \phi_1(r)$

$\sigma_w^2 = c_w^2 (\phi_1^2 - \phi_{1}^2)$
NPV of a single cash flow obtained after multiple stages

\[ v_w = c_w \phi_1(r) \ldots \phi_w(r) \]
NPV of a single cash flow obtained after multiple stages

\[ v_w = c_w \phi_1(r) \cdots \phi_w(r) \]

\[ v_w = c_w \prod_{i=1}^{w} \phi_i(r) \]
NPV of a single cash flow obtained after multiple stages

\[ \nu_w = c_w \phi_1(r) \cdots \phi_w(r) \quad \nu_w = c_w \prod_{i=1}^{w} \phi_i(r) \quad \nu_w = c_w \phi_{1,w}(r) \]
NPV of a single cash flow obtained after multiple stages

\[
\begin{align*}
\nu_w &= c_w \phi_1(r) \cdots \phi_w(r) \\
\nu_w &= c_w \prod_{i=1}^{w} \phi_i(r) \\
\nu_w &= c_w \phi_{1,w}(r)
\end{align*}
\]

- We can obtain the moments of the NPV of cash flow \( c_w \):
  - \( \mu_w = c_w \phi_{1,w}(r) \)
  - \( \sigma_w^2 = c_w^2 (\phi_{1,w}(2r) - \phi_{1,w}(r)) \)
  - ...
NPV of a single cash flow obtained after multiple stages

- The mean and variance of the distribution of time until cash flow $c_w$ is incurred is:

  $\mu_w = \sum_{i=1}^{w} \mu_i - \sum_{i=1}^{w} \sigma_i^2$,
  $\sigma_w^2 = \sum_{i=1}^{w} \sigma_i^2$.

- If stage $w$ is preceded by a sufficient number of stages, $f_1(t)$, $\ldots$, $f_{w-1}(t)$ are normally distributed with mean $\mu_w$ and variance $\sigma_w^2$.

- If $f_1(t), \ldots, f_{w-1}(t)$ are normally distributed, the NPV of cash flow $c_w$ is lognormally distributed!
NPV of a single cash flow obtained after multiple stages

\[ f_1(t) \quad 1 \quad \ldots \quad w - 1 \quad f_w(t) \quad w \]

\[ d_1 \quad d_{\ldots} \quad d_w \]

\[ c_w \]
NPV of a single cash flow obtained after multiple stages

\[ f_1(t) \]

stage 1

\[ d_1 \]

\[ s_1^2 \]

\[ \ldots \]

stage \( w - 1 \)

\[ d_{w - 1} \]

\[ s_{w - 1}^2 \]

\[ f_w(t) \]

stage \( w \)

\[ d_w \]

\[ s_w^2 \]

\[ c_w \]
NPV of a single cash flow obtained after multiple stages

• The mean and variance of the distribution of time until cash flow $c_w$ is incurred is:
  - $d_{1,w} = \sum_{i=1}^{w} d_i$
  - $s_{1,w}^2 = \sum_{i=1}^{w} s_i^2$
NPV of a single cash flow obtained after multiple stages

- The mean and variance of the distribution of time until cash flow $c_w$ is incurred is:
  - $d_{1,w} = \sum_{i=1}^{w} d_i$
  - $s_{1,w}^2 = \sum_{i=1}^{w} s_i^2$

- If stage $w$ is preceded by a sufficient number of stages, $f_{1,w}(t)$ is normally distributed with mean $d_{1,w}$ and variance $s_{1,w}^2$
The mean and variance of the distribution of time until cash flow $c_w$ is incurred is:

- $d_{1,w} = \sum_{i=1}^{w} d_i$
- $s_{1,w}^2 = \sum_{i=1}^{w} s_i^2$

If stage $w$ is preceded by a sufficient number of stages, $f_{1,w}(t)$ is normally distributed with mean $d_{1,w}$ and variance $s_{1,w}^2$.

If $f_{1,w}(t)$ is normally distributed, the NPV of cash flow $c_w$ is lognormally distributed!
Agenda

• Introduction

• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages

• General projects

• Conclusions
NPV of a serial project
NPV of a serial project

\[ v_1 = v_1 + \ldots + v_{w-1} + v_w - c_1 \]
NPV of a serial project

\[ v_1 = \sum_{i=1}^{w-1} v_i + v_w \]
NPV of a serial project

\[ \text{NPV} = \nu_1 + \ldots + \nu_{w-1} + \nu_w \]
NPV of a serial project

\[ v = v_1 + \ldots + v_{w-1} + v_w \]
NPV of a serial project

We can obtain the moments of the NPV of the serial project using exact, closed-form formula’s:
NPV of a serial project

We can obtain the moments of the NPV of the serial project using exact, closed-form formula’s:

\[
\mu_w = c_w a_1
\]

\[
\Sigma_c(w, w) = \sigma_w^2 = c_w^2 (a_2 - a^2)
\]

\[
\Sigma_c(w, x) = c_w c_x b_1 (a_2 - a^2) = c_w^{-1} c_x b_1 \Sigma_c(w, w)
\]

\[
\Gamma_c(w, w, w) = \gamma_w \sigma_w^2 = c_w^2 (a_3 - 3a_2 a_1 + 2a^3)
\]

\[
\Gamma_c(w, w, x) = c_w^{-1} c_x b_1 \Gamma_c(w, w, w)
\]

\[
\Gamma_c(w, x, x) = c_w c_x^2 (a_3 b_2 - a_2 a_1 (2b_2 + b_2 + 2a^2 b_2))
\]

\[
\Gamma_c(w, x, y) = c_w^{-1} c_x c_y h_1 \Gamma_c(w, x, x)
\]

\[
\theta_c(w, w, w, w) = \theta_w \sigma_w^4 = c_w^4 \left(a_4 - 4a_3 a_1 + 6a_2 a_1^2 - 3a_1^3 + a_1^4\right)
\]

\[
\theta_c(w, w, w, x) = c_w^{-1} c_x b_1 \theta_c(w, w, w, w)
\]

\[
\theta_c(w, w, x, x) = c_w^2 c_x^2 \left(a_4 b_2 - 2a_3 a_1 (b_2 + b_2) + a_2 a_1 (b_2 + 5b_2) - 3a_1^2 b_2\right)
\]

\[
\theta_c(w, w, x, y) = c_w^{-1} c_x h_1 \theta_c(w, w, x, x)
\]

\[
\theta_c(w, x, x, x) = c_w c_x^2 \left(a_4 b_3 - a_3 a_1 (b_3 + 3b_3 b_1) - 3a_2 a_1^2 (b_3 b_1 + 3b_3) \right.

\[
+ \left. (a_2 a_1 - a_1^2) (2b_3 b_1 + 2a^2 b_3 b_1) + (a_2 a_1 - a_1^2) (3a^3 b_1^2)\right)
\]

\[
\theta_c(w, x, y, y) = c_w^{-1} c_x c_y h_1 \theta_c(w, x, x, x)
\]
We develop a three-parameter lognormal distribution that can be used to match the mean, variance, and skewness of the true NPV distribution.
NPV of a serial project

We develop a three-parameter lognormal distribution that can be used to match the mean, variance, and skewness of the true NPV distribution.

The example below illustrates the accuracy of the three-parameter lognormal distribution ($\mathcal{L}_3$):
Agenda

• Introduction

• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages

• General projects

• Conclusions
Optimal sequence of stages

- Moments of known sequence can be obtained using exact closed-form formulas
Optimal sequence of stages

- Moments of known sequence can be obtained using exact closed-form formulas
- How to obtain the optimal sequence of a set of stages that are potentially precedence related?
Optimal sequence of stages

The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP).

The LCFDP minimizes the cost of the sequential diagnosis of a number of system components.

In the absence of precedence relations, the optimal sequence can be found in polynomial time.

Efficient algorithms are available for the general case.
• The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP)
The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP).

The LCFDP minimizes the cost of the sequential diagnosis of a number of system components.
The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP).

The LCFDP minimizes the cost of the sequential diagnosis of a number of system components.

In the absence of precedence relations, the optimal sequence can be found in polynomial time.
Optimal sequence of stages

- The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP)
- The LCFDP minimizes the cost of the sequential diagnosis of a number of system components
- In the absence of precedence relations, the optimal sequence can be found in polynomial time
- Efficient algorithms are available for the general case
Agenda

• Introduction
• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages
• General projects
• Conclusions
NPV of a general project

• Different policies can be used to schedule the 3 stages:
  – Serial
  – Early Start (ES)
  – Optimal

• If $c_3$ is obtained after stages 1 & 2, then what is discount factor?

=> Approximations may be required!

• How does $v_3$ relate to the $v_1$ & $v_2$?

What are the cross moments?

=> Approximations may be required!
NPV of a general project
Scheduling policies

• Serial policies:
  - 1 - 2 - 3
  - 1 - 3 - 2
  - 2 - 1 - 3
  - 2 - 3 - 1
  - 3 - 1 - 2
  - 3 - 2 - 1

• Early Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

• Optimal policy: Start 2. Start 1 & 3 upon completion of 2.

\[
\begin{align*}
  c_1 &= -50 \\
  c_2 &= -20 \\
  c_3 &= -10 \\
  f_1(t) &\sim \text{Exp}(1) \\
  f_2, 3(t) &\sim \text{Exp}(0.5) \\
  r &= 0.1 \\
  p &= 200
\end{align*}
\]
NPV of a general project
Scheduling policies

- Serial policies:
  - Stage 1 → Stage 2 → Stage 3
  - Stage 2 → Stage 1 → Stage 3
  - Stage 2 → Stage 3 → Stage 1
  - Stage 3 → Stage 1 → Stage 2

- Early Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

- Optimal policy: Start 2. Start 1 & 3 upon completion of 2.

\[
\begin{align*}
c_1 &= -50 \\
c_2 &= -20 \\
c_3 &= -10
\end{align*}
\]
NPV of a general project
Scheduling policies

- Serial policies:
  - 1 - 2 - 3
  - 1 - 3 - 2
  - 2 - 1 - 3
  - 2 - 3 - 1
  - 3 - 1 - 2
  - 3 - 2 - 1

- Early Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

- Optimal policy: Start 2. Start 1 & 3 upon completion of 2.

\[ c_1 = -50 \]
\[ c_2 = -20 \]
\[ c_3 = -10 \]

\[ f_1(t) \sim \text{Exp}(1) \]
\[ f_{2,3}(t) \sim \text{Exp}(0.5) \]
NPV of a general project
Scheduling policies

- **Serial policies:**
  1 - 2 - 3
  1 - 3 - 2
  2 - 1 - 3
  2 - 3 - 1
  3 - 1 - 2
  3 - 2 - 1

- **Early-Start (ES) policy:** Start 1 & 2. Start 3 upon completion of 2.

- **Optimal policy:** Start 2. Start 1 & 3 upon completion of 2.

Stage 1:
\[ c_1 = -50 \]

Stage 2:
\[ c_2 = -20 \]

Stage 3:
\[ c_3 = -10 \]

\[ f_1(t) \sim \text{Exp}(1) \]
\[ f_{2,3}(t) \sim \text{Exp}(0.5) \]
\[ p = 200 \]
NPV of a general project
Scheduling policies

- Serial policies:
  1 - 2 - 3
  1 - 3 - 2
  2 - 1 - 3
  2 - 3 - 1
  3 - 1 - 2
  3 - 2 - 1

- Early-Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

- Optimal policy: Start 2. Start 1 & 3 upon completion of 2.

\[
c_1 = -50
\]
\[
c_2 = -20
\]
\[
c_3 = -10
\]
\[
f_1(t) \sim \text{Exp}(1)
\]
\[
f_2,3(t) \sim \text{Exp}(0.5)
\]
\[
p = 200 \quad r = 0.1
\]
NPV of a general project
Scheduling policies

• **Serial policies:**
  - 1-2-3
  - 1-3-2
  - 2-1-3
  - 2-3-1
  - 3-1-2
  - 3-2-1

---

- $c_1 = -50$
- $c_2 = -20$
- $c_3 = -10$

- $f_1(t) \sim \text{Exp}(1)$
- $f_{2,3}(t) \sim \text{Exp}(0.5)$
- $p = 200$, $r = 0.1$
NPV of a general project
Scheduling policies

• Serial policies:
  - 1-2-3
  - 1-3-2
  - 2-1-3
  - 2-3-1
  - 3-1-2
  - 3-2-1

• Early-Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

\[ c_1 = -50 \]
\[ c_2 = -20 \]
\[ c_3 = -10 \]

\[ f_1(t) \sim \text{Exp}(1) \]
\[ f_{2,3}(t) \sim \text{Exp}(0.5) \]
\[ p = 200 \quad r = 0.1 \]
NPV of a general project
Scheduling policies

- **Serial policies:**
  - 1-2-3
  - 1-3-2
  - 2-1-3
  - 2-3-1
  - 3-1-2
  - 3-2-1

- **Early-Start (ES) policy:** Start 1 & 2. Start 3 upon completion of 2.

- **Optimal policy:** Start 2. Start 1 & 3 upon completion of 2.

\[
c_1 = -50 \]
\[
c_2 = -20 \]
\[
c_3 = -10 \]

\[
f_1(t) \sim \text{Exp}(1)\]
\[
f_{2,3}(t) \sim \text{Exp}(0.5)\]

\[
p = 200 \quad r = 0.1\]
NPV of a general project

Early-Start policy

- When do we incur the payoff?
  - After stage 1?
  - After stage 2&3?

- What discount factor do we use?
  - \( \phi_1(r) \)
  - \( \phi_{2,3}(r) \)

- There no longer exists a fixed sequence/the sequence is probabilistic \( \Rightarrow \) Approximations are required!

\[
\begin{align*}
  c_1 &= -50 \\
  c_2 &= -20 \\
  c_3 &= -10 \\
  f_1(t) &\sim \text{Exp}(1) \\
  f_{2,3}(t) &\sim \text{Exp}(0.5) \\
  p &= 200 \\
  r &= 0.1
\end{align*}
\]
NPV of a general project
Early-Start policy

• When do we incur the payoff?
  – After stage 1?
  – After stage 2&3?

$c_1 = -50$

$c_2 = -20$
$c_3 = -10$

$f_1(t) \sim \text{Exp}(1)$
$f_{2,3}(t) \sim \text{Exp}(0.5)$
$p = 200$  $r = 0.1$
NPV of a general project
Early-Start policy

- When do we incur the payoff?
  - After stage 1?
  - After stage 2&3?
- What discount factor do we use?
  - $\phi_1(r)$
  - $\phi_{2,3}(r)$

$c_1 = -50$
$c_2 = -20$
$c_3 = -10$

$f_1(t) \sim \text{Exp}(1)$
$f_{2,3}(t) \sim \text{Exp}(0.5)$
$p = 200$  $r = 0.1$
NPV of a general project
Early-Start policy

• When do we incur the payoff?
  – After stage 1?
  – After stage 2&3?

• What discount factor do we use?
  – $\phi_1(r)$
  – $\phi_{2,3}(r)$

• There no longer exists a fixed sequence/the sequence is probabilistic

$c_1 = -50$
$c_2 = -20$
$c_3 = -10$

$f_1(t) \sim \text{Exp}(1)$
$f_{2,3}(t) \sim \text{Exp}(0.5)$
$p = 200 \quad r = 0.1$
NPV of a general project

Early-Start policy

- When do we incur the payoff?
  - After stage 1?
  - After stage 2&3?

- What discount factor do we use?
  - $\phi_1(r)$
  - $\phi_{2,3}(r)$

- There no longer exists a fixed sequence/the sequence is probabilistic
  \[ f_1(t) \sim \text{Exp}(1) \]
  \[ f_{2,3}(t) \sim \text{Exp}(0.5) \]

\[ p = 200 \quad r = 0.1 \]
NPV of a general project

Optimal policy

- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel
- What discount factor do we use?
  - \( \phi_2(\tau) \phi_1(\tau) \)
  - \( \phi_2(\tau) \phi_3(\tau) \)
- The payoff is obtained after the maximum duration of stages 1 & 3!
  - We need to determine the discount factor for this maximum duration
  - If this is not possible, approximations are required!

\[
c_1 = -50
\]

\[
c_2 = -20
\]

\[
c_3 = -10
\]

\[
f_1(t) \sim \text{Exp}(1)
\]

\[
f_{2,3}(t) \sim \text{Exp}(0.5)
\]

\[
p = 200 \quad r = 0.1
\]
**NPV of a general project**

**Optimal policy**

- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel.

- What discount factor do we use?
  - $\phi_2(r)$
  - $\phi_1(r)$
  - $\phi_3(r)$

- The payoff is obtained after the maximum duration of stages 1 & 3.

  \[ \text{We need to determine the discount factor for this maximum duration.} \]

  \[ \text{If this is not possible, approximations are required!} \]

- Stage 1: $c_1 = -50$
- Stage 2: $c_2 = -20$
- Stage 3: $c_3 = -10$

- $f_1(t) \sim Exp(1)$
- $f_{2,3}(t) \sim Exp(0.5)$

- $p = 200$  $r = 0.1$
Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel.

What discount factor do we use?
- $\phi_2(r) \phi_1(r)$
- $\phi_2(r) \phi_3(r)$

Stage 1
- $c_1 = -50$

Stage 2
- $c_2 = -20$

Stage 3
- $c_3 = -10$

$f_1(t) \sim \text{Exp}(1)$

$f_{2,3}(t) \sim \text{Exp}(0.5)$

$p = 200 \quad r = 0.1$
NPV of a general project

Optimal policy

• Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel

• What discount factor do we use?
  – \( \phi_2(r) \phi_1(r) \)
  – \( \phi_2(r) \phi_3(r) \)

• The payoff is obtained after the maximum duration of stages 1 & 3!

\[
\begin{align*}
c_1 &= -50 \\
c_2 &= -20 \\
c_3 &= -10
\end{align*}
\]

\[
\begin{align*}
f_1(t) &\sim \text{Exp}(1) \\
f_{2,3}(t) &\sim \text{Exp}(0.5)
\end{align*}
\]

\[p = 200 \quad r = 0.1\]
NPV of a general project

Optimal policy

- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel
- What discount factor do we use?
  - $\phi_2(r) \phi_1(r)$
  - $\phi_2(r) \phi_3(r)$
- The payoff is obtained after the maximum duration of stages 1 & 3!
  ⇒ We need to determine the discount factor for this maximum distribution

\[
c_1 = -50
\]
\[
c_2 = -20
\]
\[
c_3 = -10
\]
\[
f_1(t) \sim Exp(1)
\]
\[
f_{2,3}(t) \sim Exp(0.5)
\]
\[
p = 200 \quad r = 0.1
\]
NPV of a general project

Optimal policy

- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel
- What discount factor do we use?
  - $\phi_2(r) \phi_1(r)$
  - $\phi_2(r) \phi_3(r)$
- The payoff is obtained after the maximum duration of stages 1 & 3!

$\Rightarrow$ We need to determine the discount factor for this maximum distribution

$\Rightarrow$ If this is not possible, approximations are required!

\[c_1 = -50\]
\[c_2 = -20\]
\[c_3 = -10\]

\[f_1(t) \sim \text{Exp}(1)\]
\[f_{2,3}(t) \sim \text{Exp}(0.5)\]

\[p = 200\]
\[r = 0.1\]
NPV of a general project

The example below illustrates the accuracy of the three-parameter lognormal distribution ($\mathcal{L}_3$) for the ES and the optimal policy:
Agenda

• Introduction
• Serial projects:
  – Single cash flow after a single stage
  – Single cash flow after multiple stages
  – NPV of a serial project
  – Optimal sequence of stages
• General projects
• Conclusions
Conclusion

• We obtain exact, closed-form expressions for the moments of the NPV of serial projects.
• The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution.
• The optimal sequence of stages can be found efficiently.
• The eNPV of a general project can be obtained using exact, closed-form expressions.
• Higher moments & the distribution of the NPV of a general project can be approximated.
Conclusion

- We obtain exact, closed-form expressions for the moments of the NPV of serial projects.
- The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution.
- The optimal sequence of stages can be found efficiently.
- The eNPV of a general project can be obtained using exact, closed-form expressions.
- Higher moments and the distribution of the NPV of a general project can be approximated.
Conclusion

• We obtain exact, closed-form expressions for the moments of the NPV of serial projects
• The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution
Conclusion

- We obtain exact, closed-form expressions for the moments of the NPV of serial projects
- The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution
- The optimal sequence of stages can be found efficiently
- The $eNPV$ of a general project can be obtained using exact, closed-form expressions
- Higher moments & the distribution of the NPV of a general project can be approximated
Conclusion

• We obtain exact, closed-form expressions for the moments of the NPV of serial projects
• The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution
• The optimal sequence of stages can be found efficiently
• The eNPV of a general project can be obtained using exact, closed-form expressions
Conclusion

• We obtain exact, closed-form expressions for the moments of the NPV of serial projects
• The distribution of the NPV of a serial project can be approximated accurately using a three-parameter lognormal distribution
• The optimal sequence of stages can be found efficiently
• The eNPV of a general project can be obtained using exact, closed-form expressions
• Higher moments & the distribution of the NPV of a general project can be approximated
TIME FOR QUESTIONS?
Vive la Mannschaft