Some recent advances in project scheduling

Stefan Creemers

(June 27, 2018)

IÉSEG SCHOOL OF MANAGEMENT

KATHOLIEKE UNIVERSITEIT LEUVEN

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

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?

Who?

Activities

Resources

Project scheduling: important concepts

What? **Who?** **When?**

Activities **Resources**

Project scheduling: important concepts

What?

Activities

Who?

Resources

When?

Schedule/Policy

Project scheduling: important concepts

What?

Activities

Who?

Resources

When?

Schedule/Policy

Why?

Project scheduling: important concepts

What?	**Who?**	**When?**	**Why?**
			
Activities	**Resources**	**Schedule/Policy**	**Makespan/NPV…**

Project scheduling problems we'll consider today

Project scheduling problems we'll consider today

- Minimize makespan:

Project scheduling problems we'll consider today

- Minimize makespan:
  - Deterministic activity durations:

Project scheduling problems we'll consider today

- Minimize makespan:
  - Deterministic activity durations:
    - No preemption: **RCPSP**

Project scheduling problems we'll consider today

- Minimize makespan:
  - Deterministic activity durations:
    - No preemption: **RCPSP**
    - Preemption: **PRCPSP**

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: **PSRCPSP**
- Maximize NPV
  - Stochastic activity durations: **SNPV**

Project scheduling problems we'll consider today

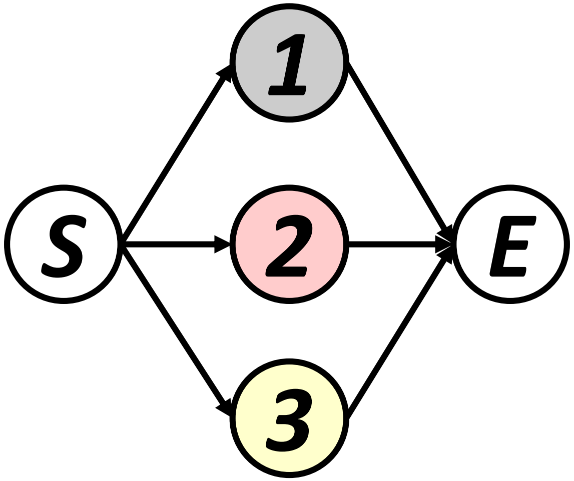
- Minimize makespan:
  - Deterministic activity durations:
    - No preemption: **RCPSP**
    - Preemption: **PRCPSP**
  - Stochastic activity durations:
    - No preemption: **SRCPSP**
    - Preemption: **PSRCPSP**
- Maximize NPV
  - Stochastic activity durations: **SNPV**
- All these problems are NP-hard!

RCPSP

(Resource-Constrained Project Scheduling Problem)

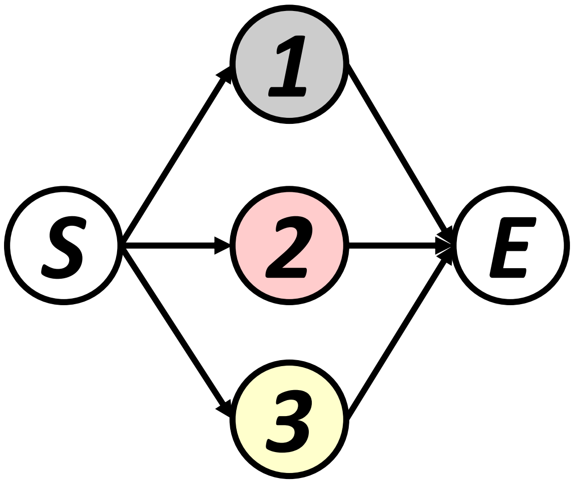
RCPSP

(Resource-Constrained Project Scheduling Problem)



RCPSP

(Resource-Constrained Project Scheduling Problem)

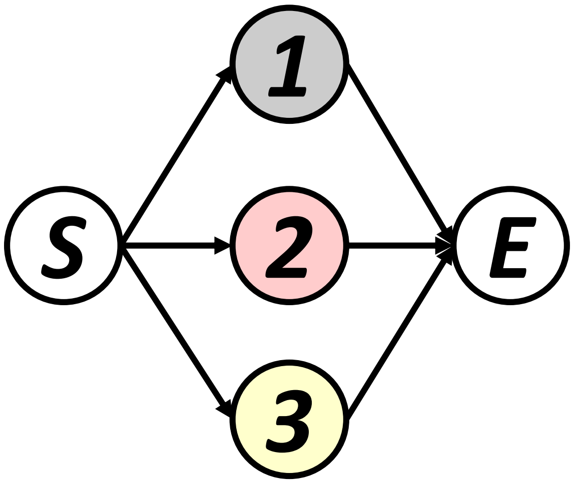


ACT	DUR	RESOURCE USE
1	2	1
2	2	1
3	2	1

Resource availability: 2

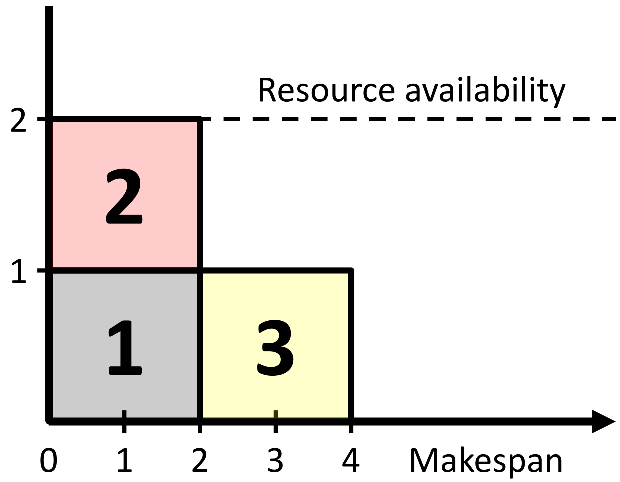
RCPSP

(Resource-Constrained Project Scheduling Problem)



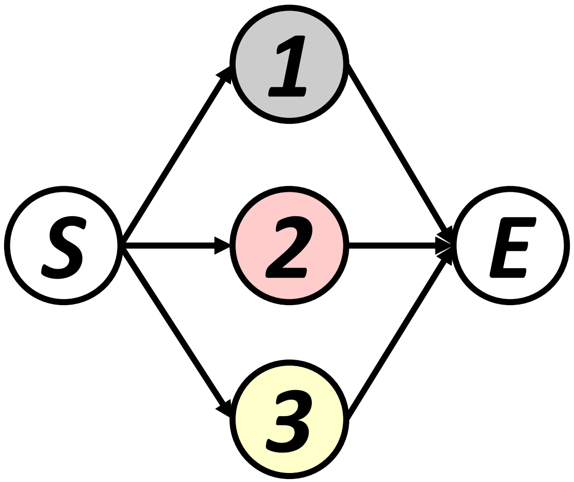
ACT	DUR	RESOURCE USE
1	2	1
2	2	1
3	2	1

Resource availability: 2



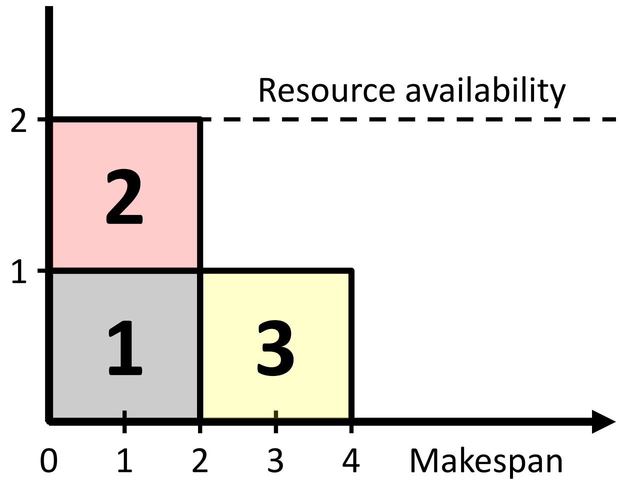
PRCPSP

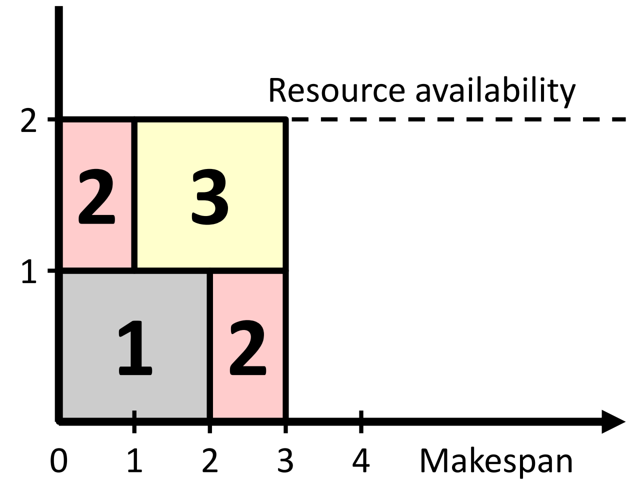
(Preemptive Resource-Constrained Project Scheduling Problem)



ACT	DUR	RESOURCE USE
1	2	1
2	2	1
3	2	1

Resource availability: 2



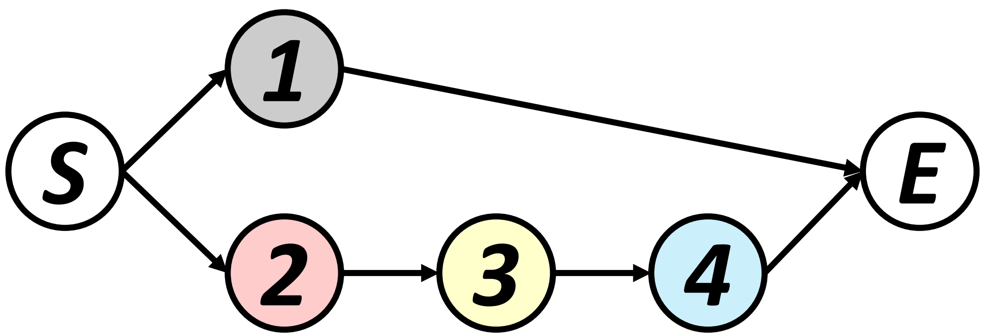


SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)

SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)

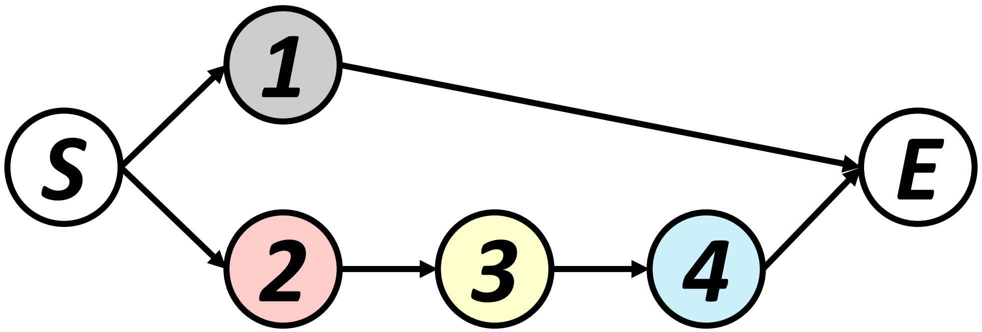


ACT	DUR	RESOURCE USE
1	4	1
2	{2,4}	1
3	2	2
4	2	1

Resource availability: 2

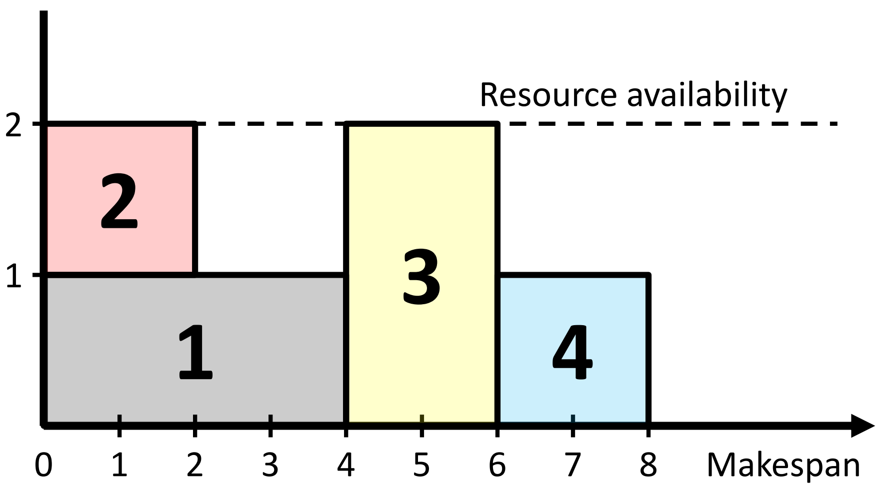
SRCPSP

(Stochastic Resource-Constrained Project Scheduling Problem)



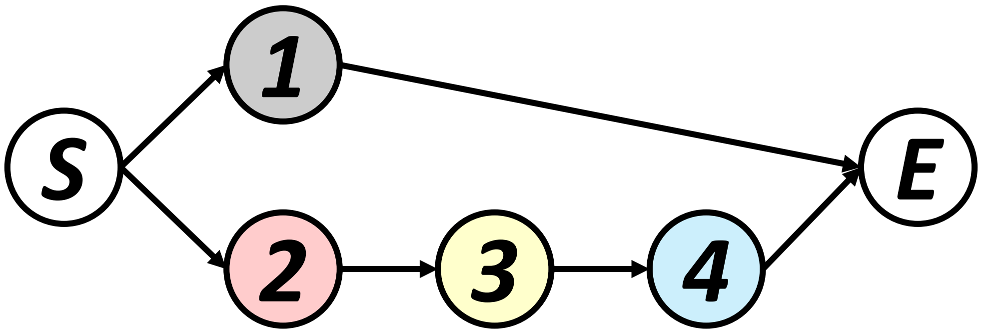
ACT	DUR	RESOURCE USE
1	4	1
2	{2,4}	1
3	2	2
4	2	1

Resource availability: 2



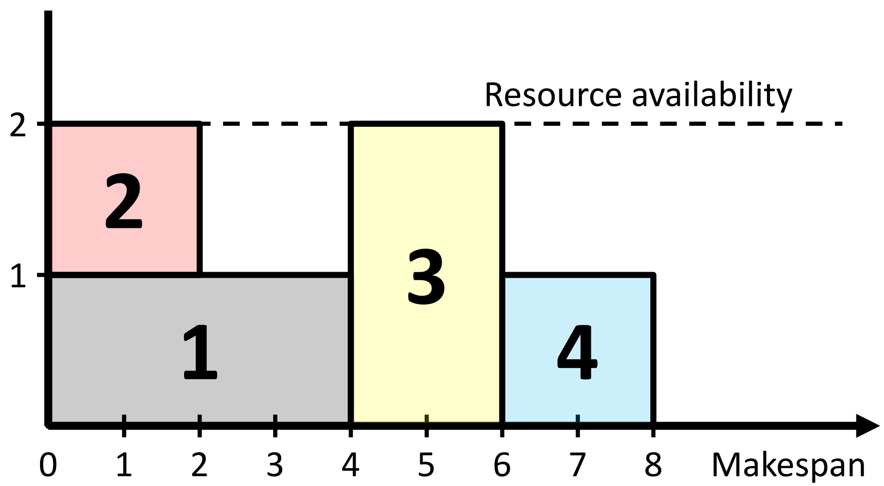
SRCPSP

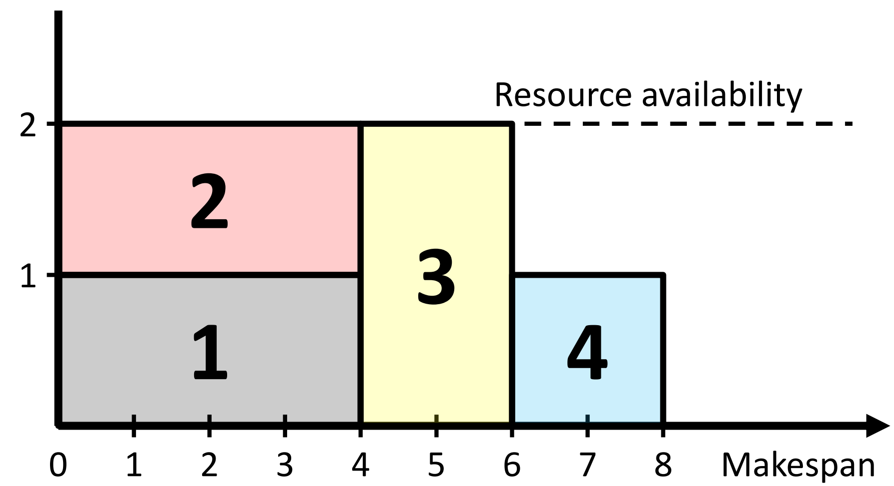
(Stochastic Resource-Constrained Project Scheduling Problem)



ACT	DUR	RESOURCE USE
1	4	1
2	{2,4}	1
3	2	2
4	2	1

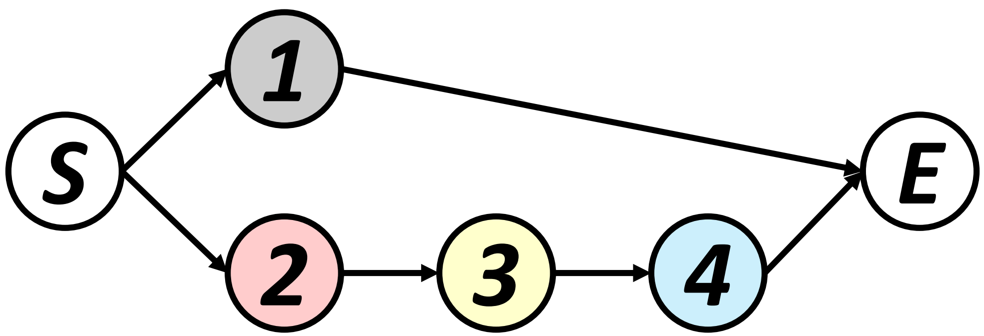
Resource availability: 2





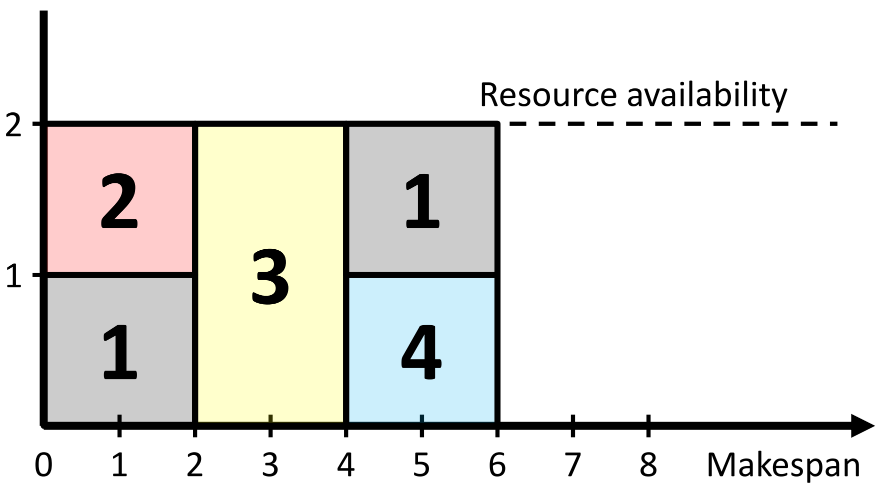
PSRCPSP

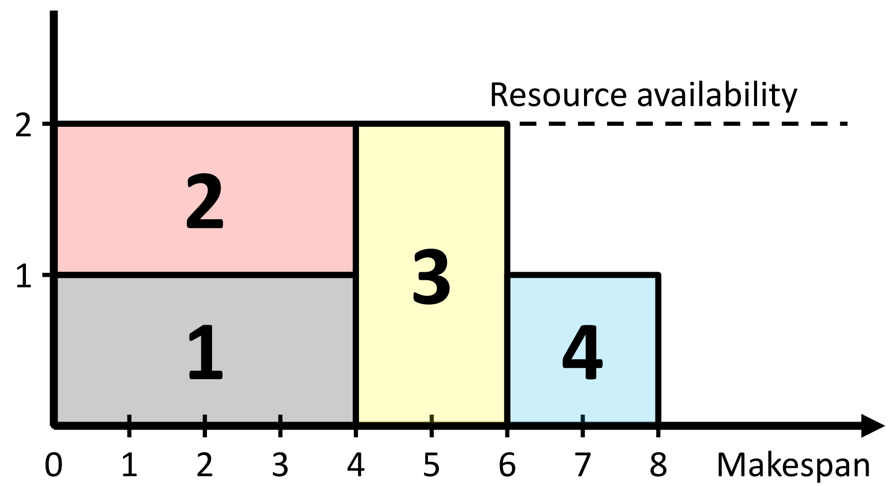
(Preemptive Stochastic Resource-Constrained Project Scheduling Problem)



ACT	DUR	RESOURCE USE
1	4	1
2	{2,4}	1
3	2	2
4	2	1

Resource availability: 2



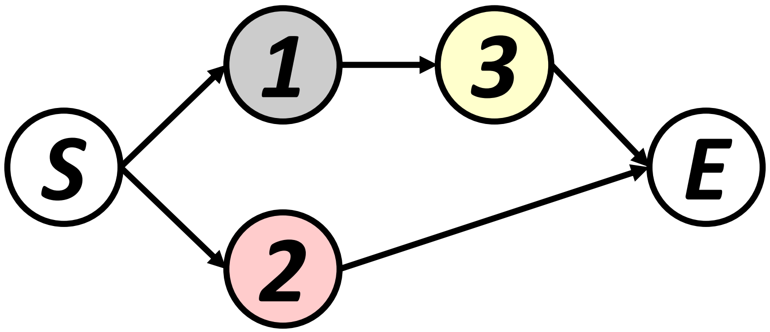


SNPV

(Stochastic expected NPV maximization problem)

SNPV

(Stochastic expected NPV maximization problem)



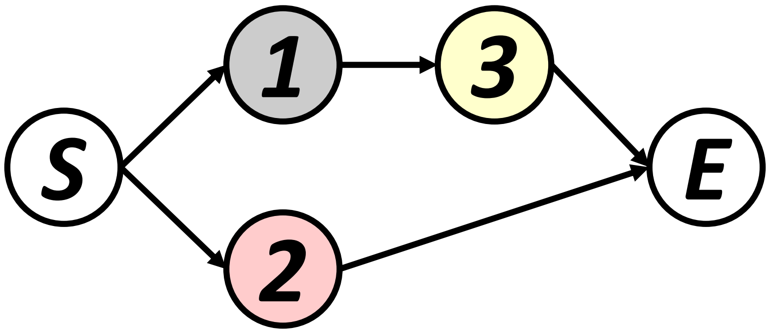
ACT	DUR	COST
1	4	0
2	{2,4}	0
3	1	-5

Discount rate: 10%

Project payoff: 10

SNPV

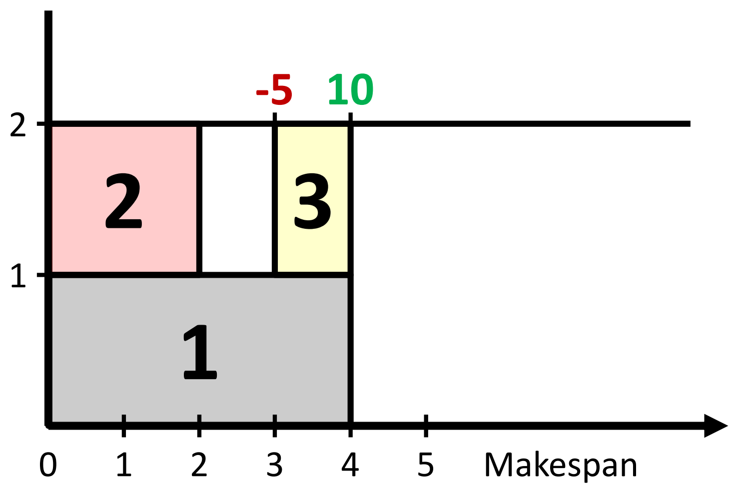
(Stochastic expected NPV maximization problem)



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

Discount rate: 10%

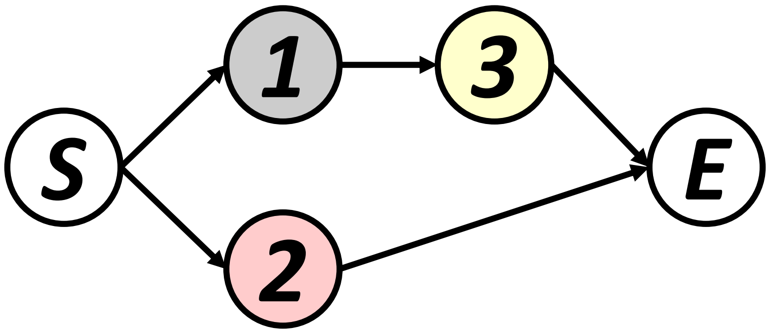
Project payoff: 10



NPV = 3.00

SNPV

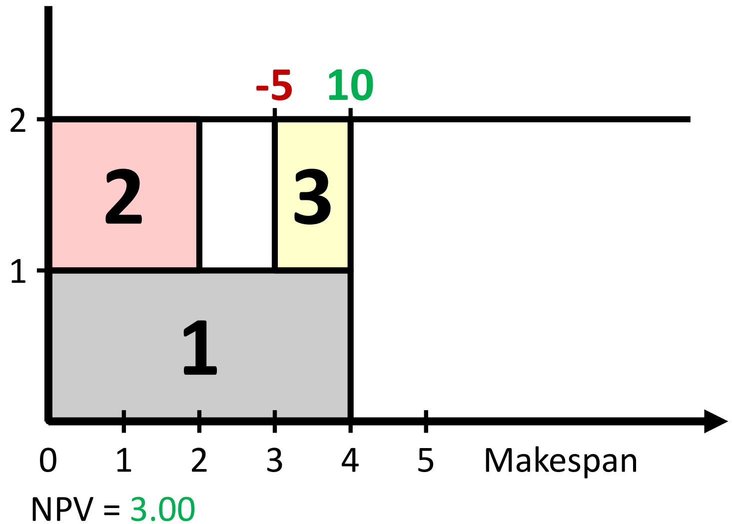
(Stochastic expected NPV maximization problem)

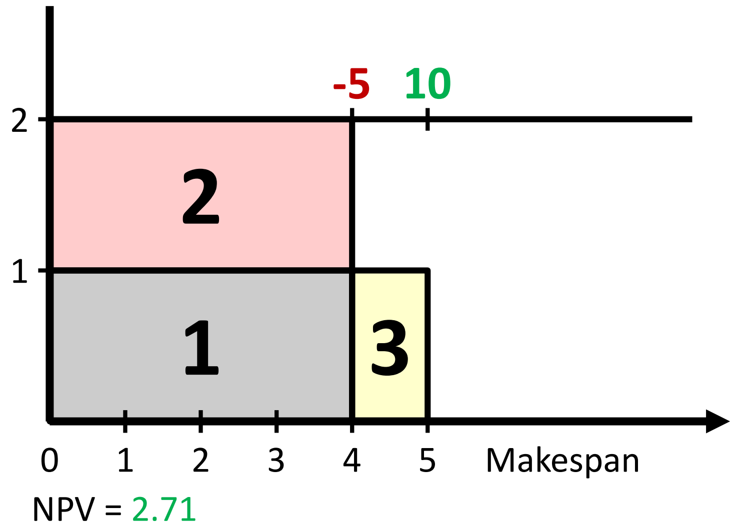


ACT	DUR	COST
1	4	0
2	{2,4}	0
3	1	-5

Discount rate: 10%

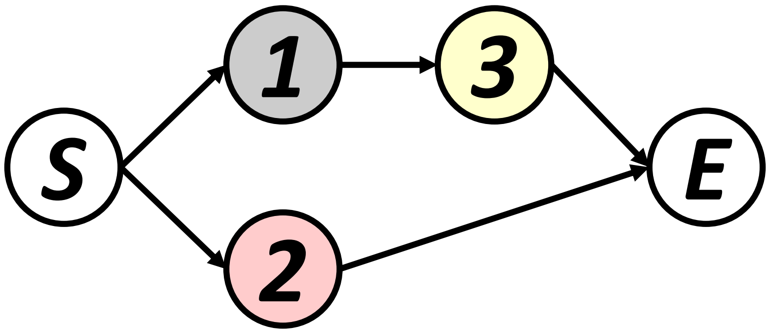
Project payoff: 10





SNPV

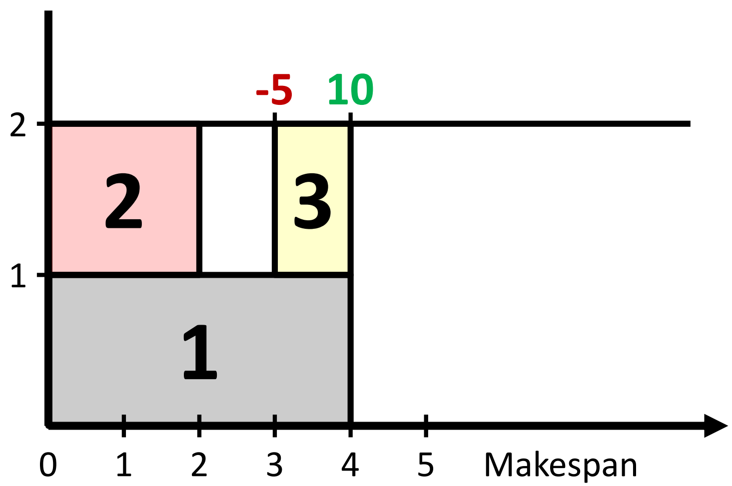
(Stochastic expected NPV maximization problem)



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

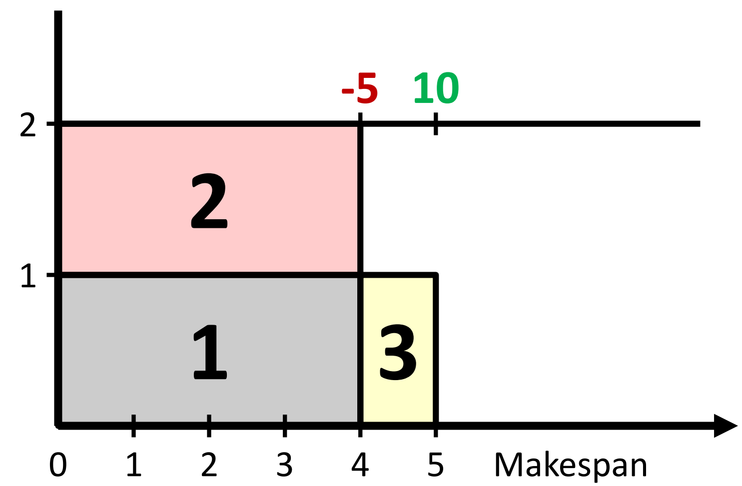
Discount rate: 10%

Project payoff: 10



NPV = 3.00

eNPV = 2.86



NPV = 2.71

THE RCPSP

The RCPSP: Facts & figures

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

1959

Bowman (MIT): first optimal solution

The RCPSP: A brief (incomplete) timeline

1959 1983

Blazewicz (Poznan): proof that RCPSP is NP complete

Bowman (MIT): first optimal solution

The RCPSP: A brief (incomplete) timeline

1992

1959 1983 1998

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

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, 1998 — Demeulemeester (KU Leuven): current state-of-the-art

2000 — Age of heuristics

The RCPSP: A brief (incomplete) timeline



1992

1959 1983 1998 2000 2006

Age of MILP solvers

Age of heuristics

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

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 / 1998 — Demeulemeester (KU Leuven): current state-of-the-art

2000 — Age of heuristics

2006 — Age of MILP solvers

2018 — Creemers (IESEG): new state-of-the-art

The RCPSP: new approach

The RCPSP: new approach

- Exact approach

The RCPSP: new approach

- Exact approach
- Work in progress

The RCPSP: new approach

- Exact approach
- Work in progress
- Preliminary results:

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

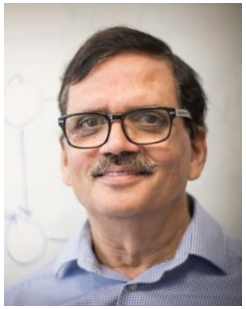
Agenda

- 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

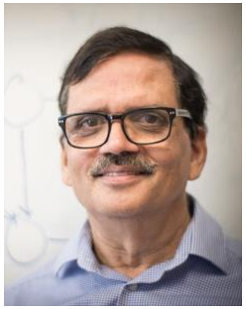
Agenda

- 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)

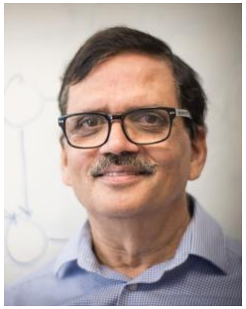


Kulkarni & Adlakha (1986)



- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986

Kulkarni & Adlakha (1986)

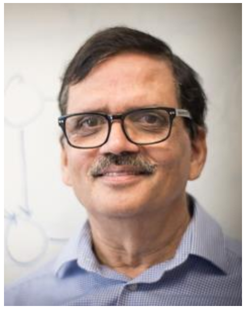


- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986
- 208 citations

Kulkarni & Adlakha (1986)

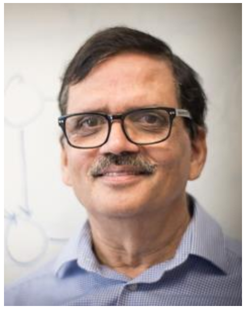
- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986
- 208 citations
- First to study Markovian PERT networks

Kulkarni & Adlakha (1986)



- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986
- 208 citations
- First to study Markovian PERT networks
- Use of a CTMC to model a network

Kulkarni & Adlakha (1986)



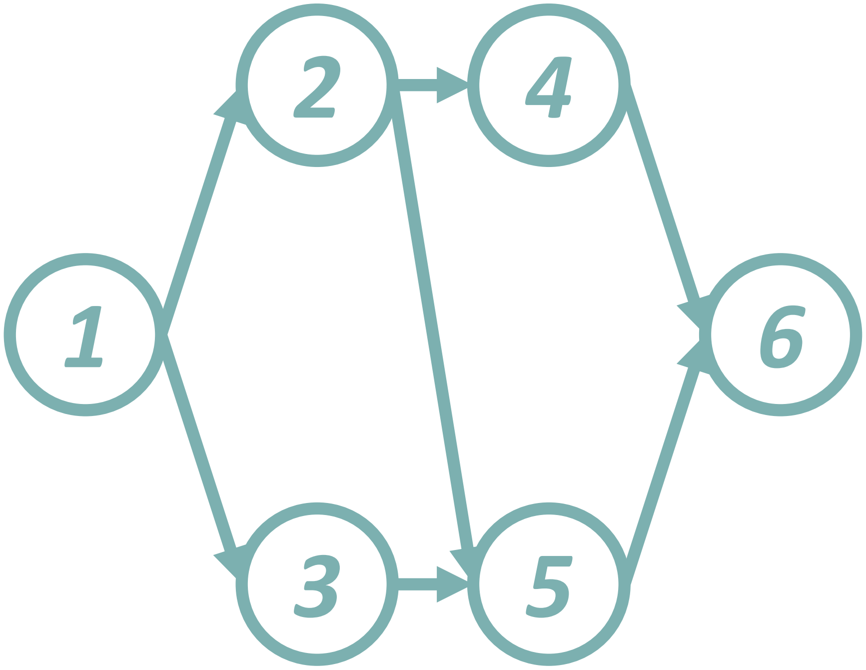
- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986
- 208 citations
- First to study Markovian PERT networks
- Use of a CTMC to model a network
- The states of the CTMC are defined by three sets: idle, ongoing, & finished activities

Kulkarni & Adlakha (1986)

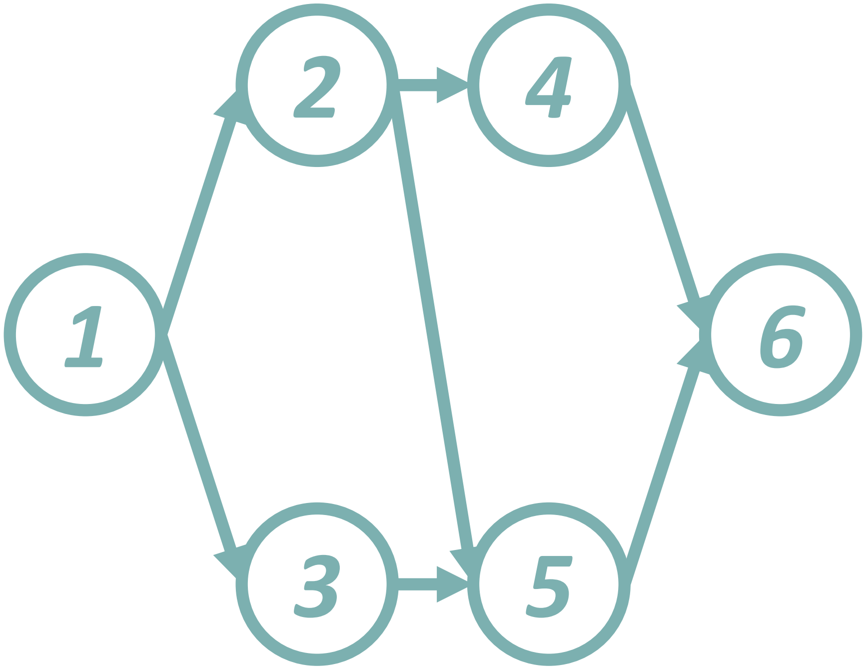
- Markov and Markov-Regenerative PERT Networks, *Operations Research*, 1986
- 208 citations
- First to study Markovian PERT networks
- Use of a CTMC to model a network
- The states of the CTMC are defined by three sets: idle, ongoing, & finished activities

$\Rightarrow$For a project with n activities there are up to 3^n states!

Example: State space

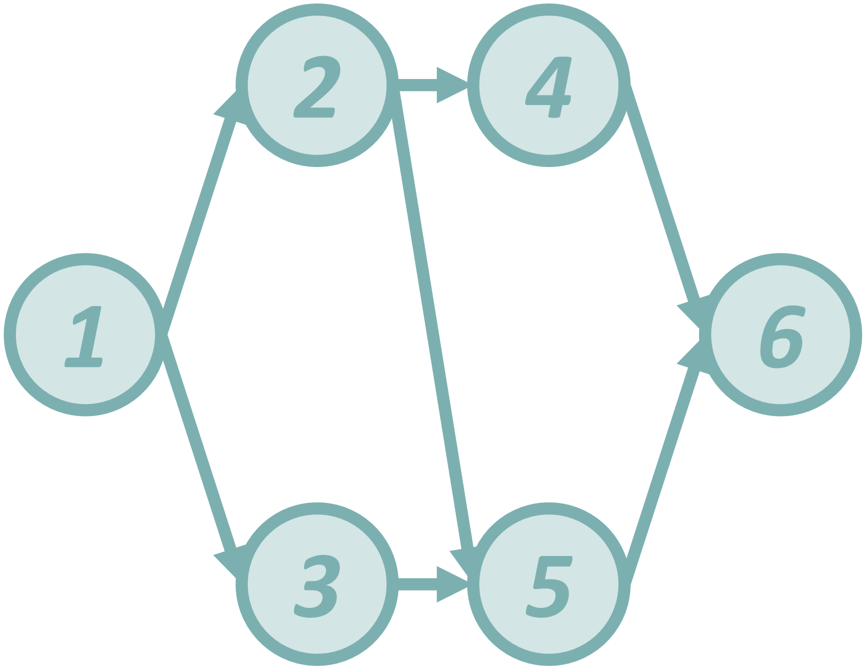


Example: State space



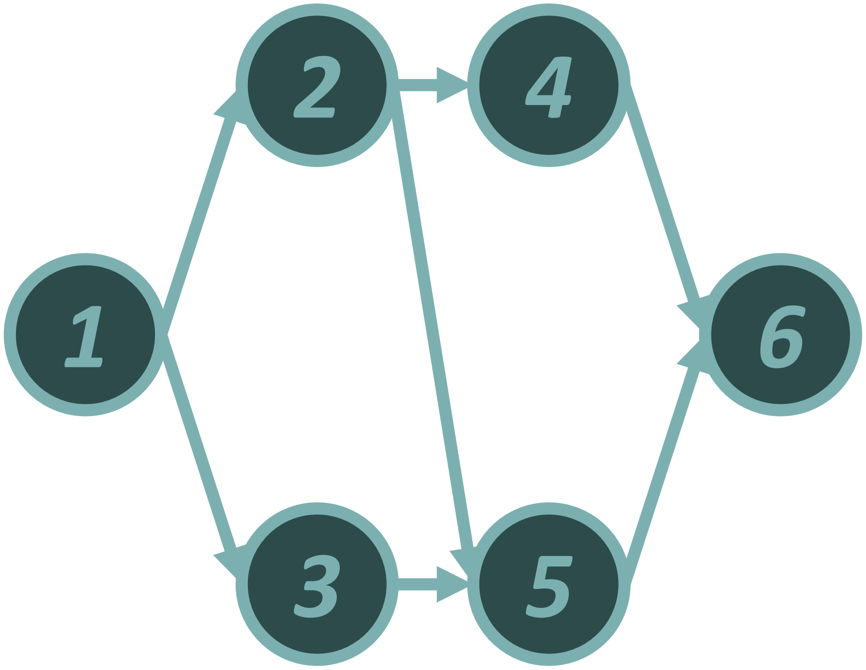
- An activity ***j*** is either:
  - Idle (θ_j=0)

Example: State space



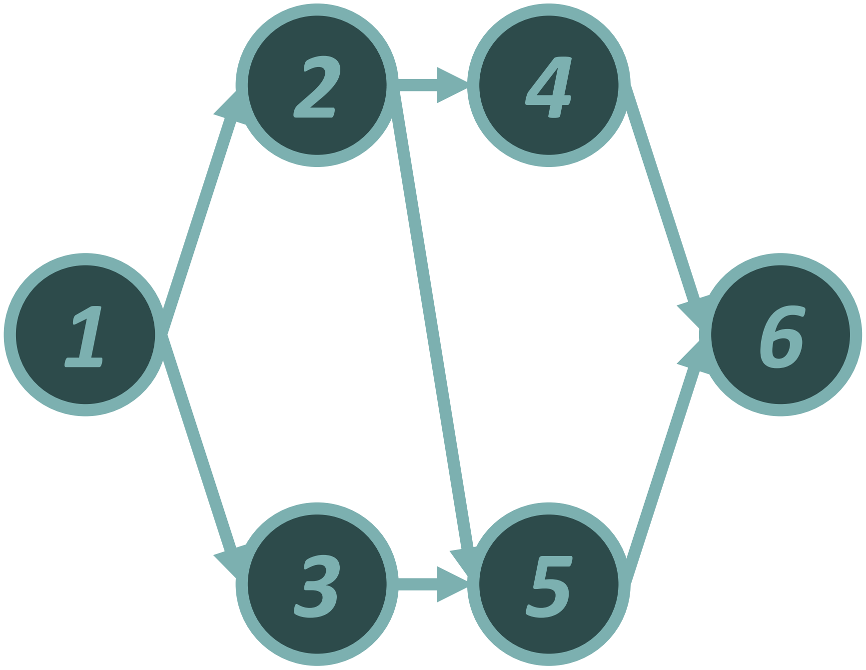
- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Ongoing (θ_j=1)

Example: State space



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

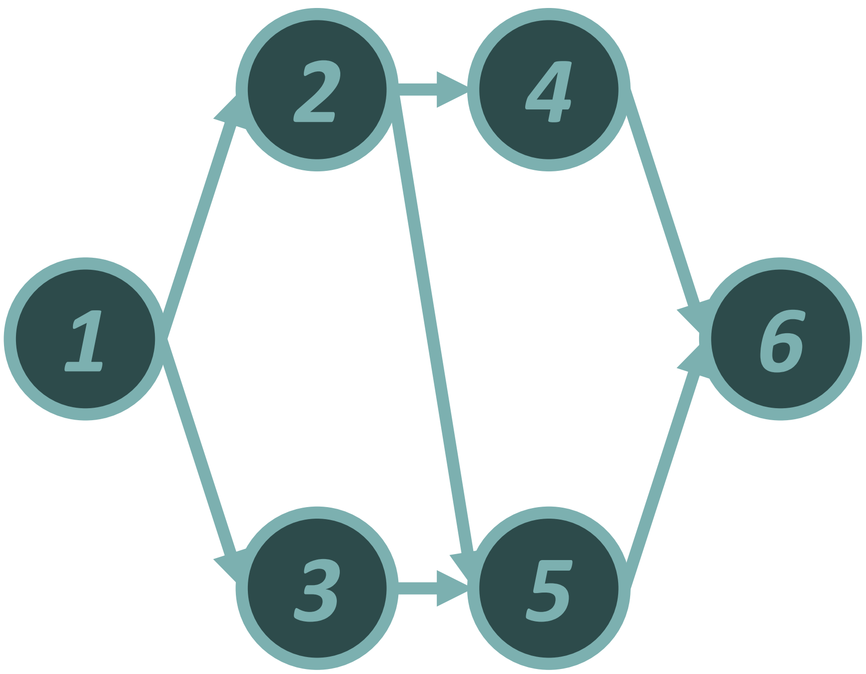
Example: State space



- An activity $\boldsymbol{j}$ is either:
  - Idle (θ_j=0)
  - Ongoing (θ_j=1)
  - Finished (θ_j=2)
- The state of the system is represented by a vector:

$\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots \theta_n\}$

Example: State space

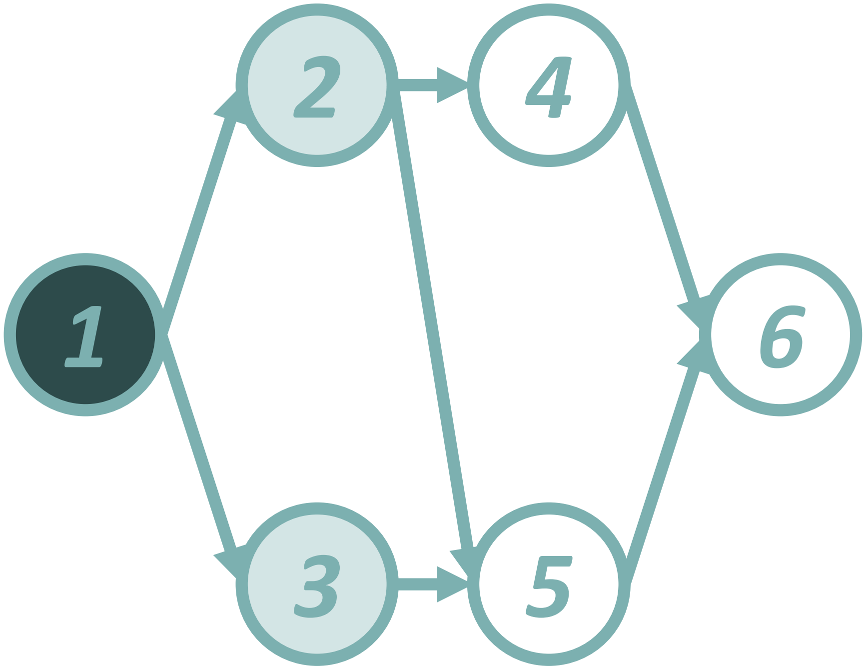


- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Ongoing (θ_j=1)
  - Finished (θ_j=2)
- The state of the system is represented by a vector:

$\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots\ \theta_n\}$

- Up to 3^n = 729 states

Example: State space



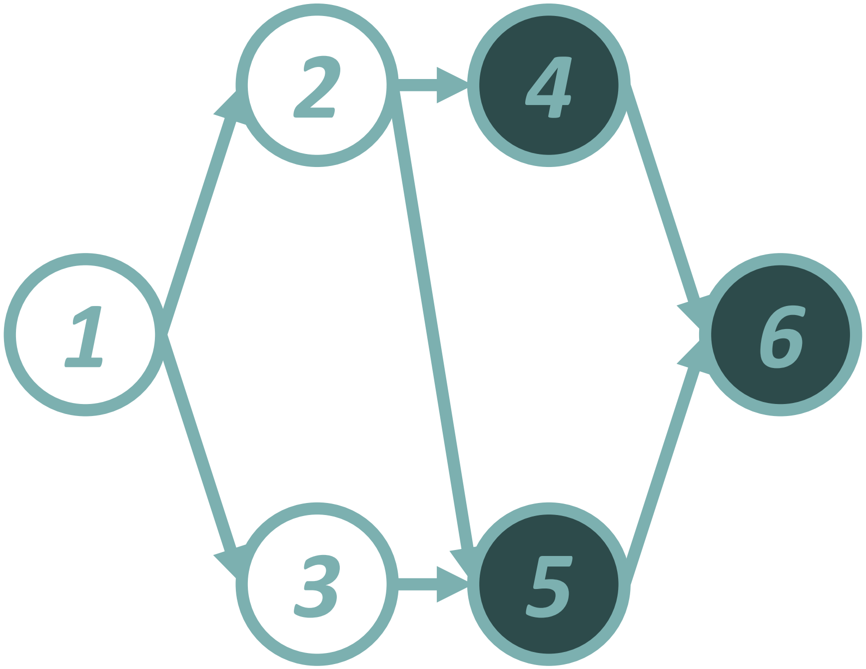
- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Ongoing (θ_j=1)
  - Finished (θ_j=2)
- The state of the system is represented by a vector:

$\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots \theta_n\}$

- Up to 3^n = 729 states
- Example feasible state:

$\boldsymbol{\theta} = \{2,1,1,0,0,0\}$

Example: State space



- An activity j is either:
  - Idle (θ_j=0)
  - Ongoing (θ_j=1)
  - Finished (θ_j=2)
- The state of the system is represented by a vector:

$\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots \theta_n\}$

- Up to 3^n = 729 states
- Example feasible state:

$\boldsymbol{\theta} = \{2,1,1,0,0,0\}$

- Example Infeasible state:

$\boldsymbol{\theta} = \{0,0,0,2,2,2\}$

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

$\Rightarrow$ up to 2^n states (instead of 3^n 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

$\Rightarrow$ up to 2^n states (instead of 3^n states)

$\Rightarrow$ Huge reduction in memory requirements (= THE bottleneck for CTMC of Kulkarni & Adlakha)

New CTMC

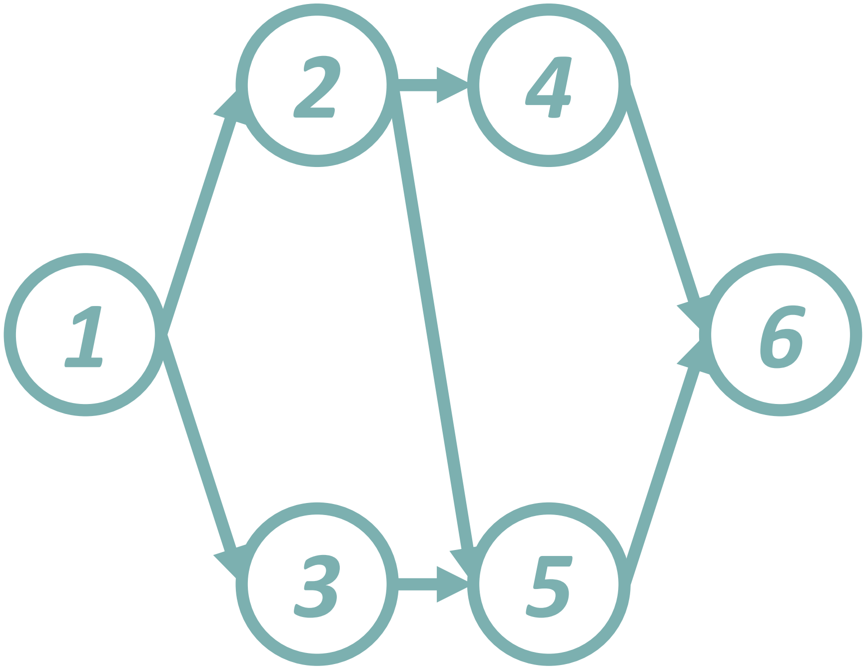
- We are the first to introduce a new CTMC since the CTMC of Kulkarni & Adlakha that was published in 1986
- In this new CTMC, states are defined by the set of finished activities

$\Rightarrow$ up to 2^n states (instead of 3^n states)

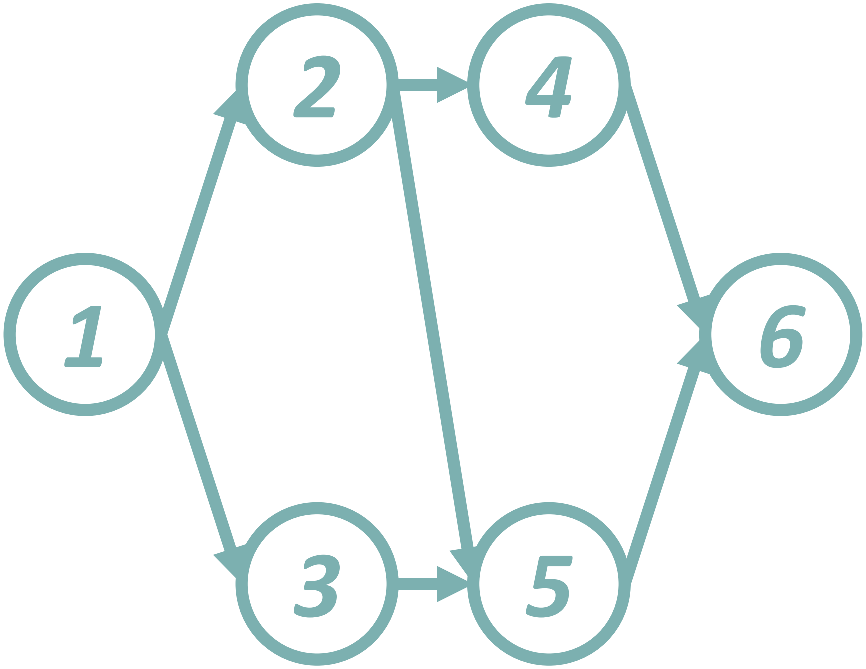
$\Rightarrow$ Huge reduction in memory requirements (= THE bottleneck for CTMC of Kulkarni & Adlakha)

- A potential "drawback" is that the new CTMC allows activities to be preempted

Example: State space

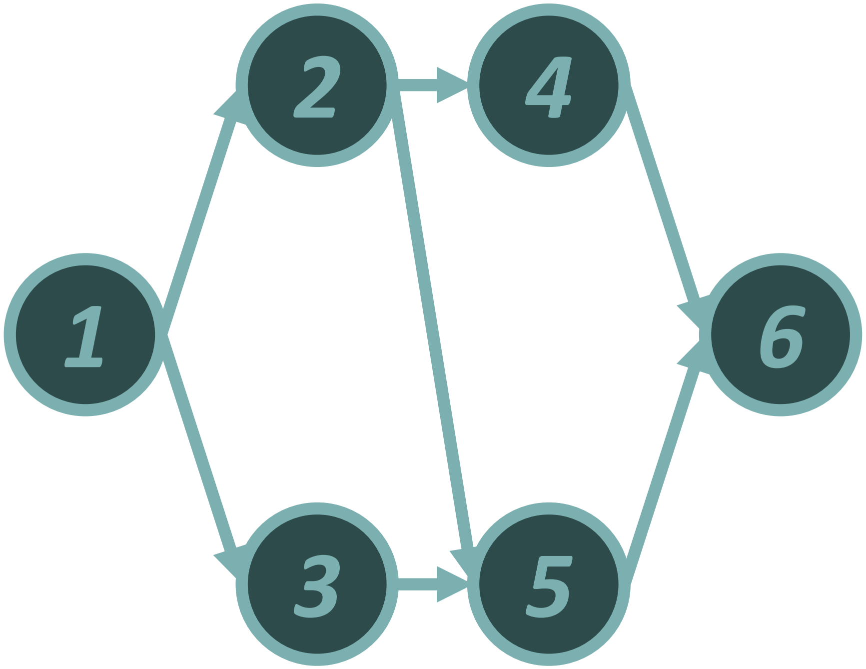


Example: State space



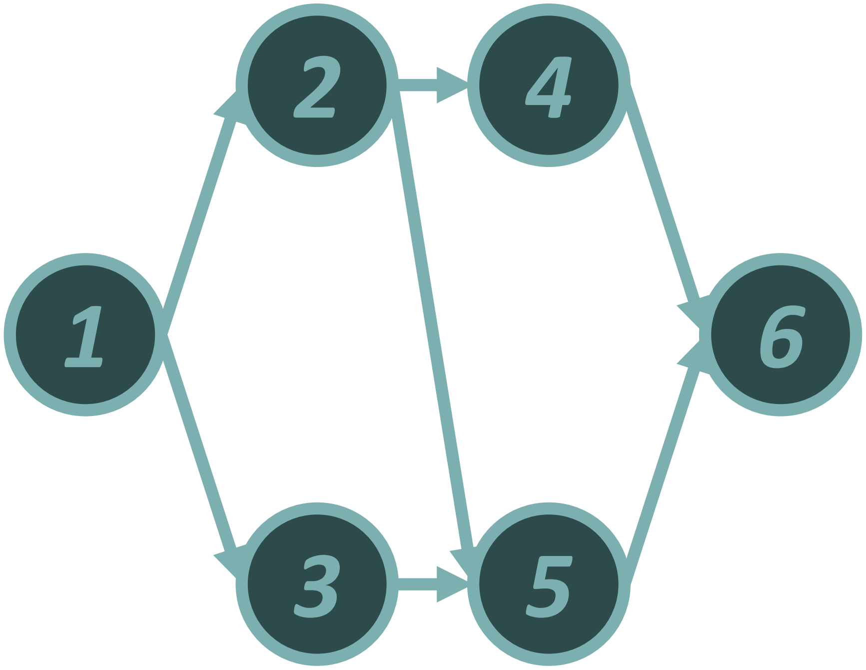
- An activity ***j*** is either:
  - Idle (θ_j=0)

Example: State space



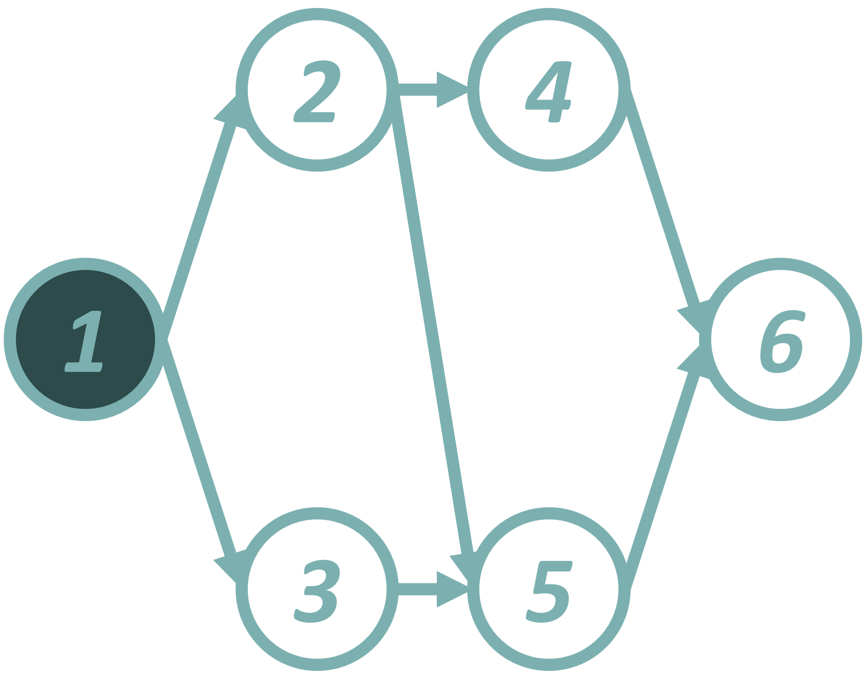
- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Finished (θ_j=1)

Example: State space



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

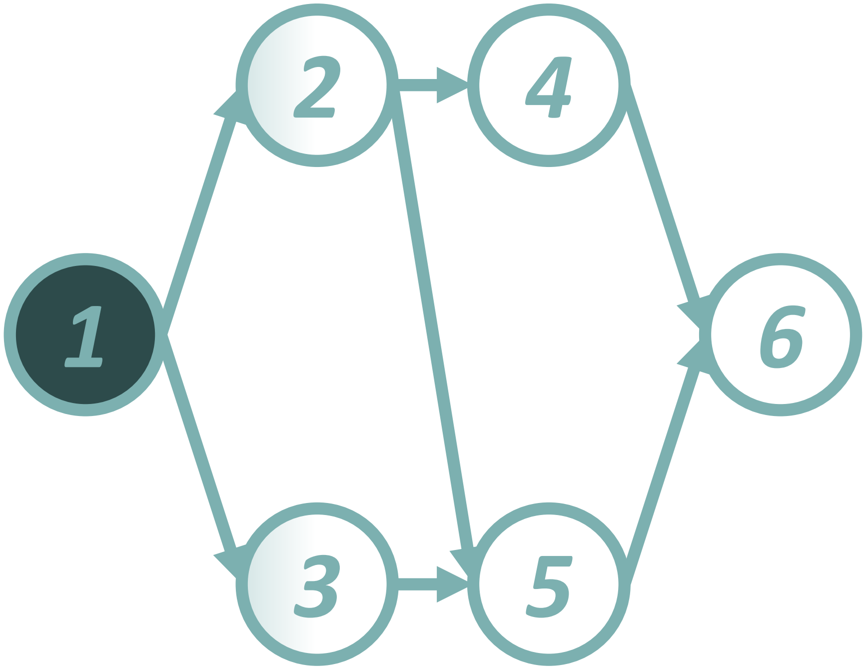
Example: State space



- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Finished (θ_j=1)
- Up to 2^n = 64 states
- Example feasible state:

$\boldsymbol{\theta}$ = {1,0,0,0,0,0}

Example: State space

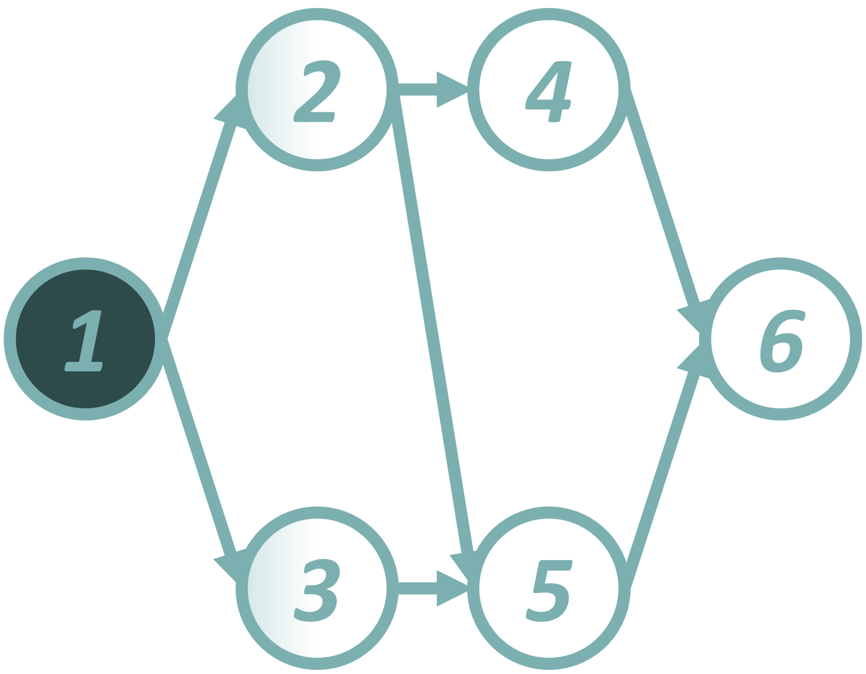


- An activity ***j*** is either:
  - Idle (θ_j=0)
  - Finished (θ_j=1)
- Up to 2^n = 64 states
- Example feasible state:

$\boldsymbol{\theta}$ = *{1,0,0,0,0,0}*

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

Example: State space

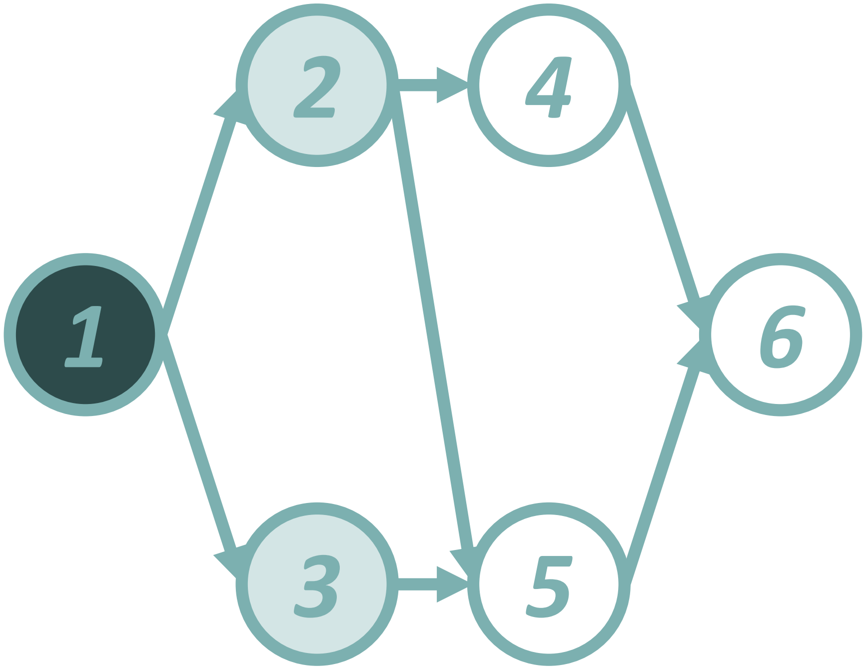


- An activity j is either:
  - Idle (θ_j=0)
  - Finished (θ_j=1)
- Up to 2^n = 64 states
- Example feasible state:

$\boldsymbol{\theta}$ = $\{1,0,0,0,0,0\}$

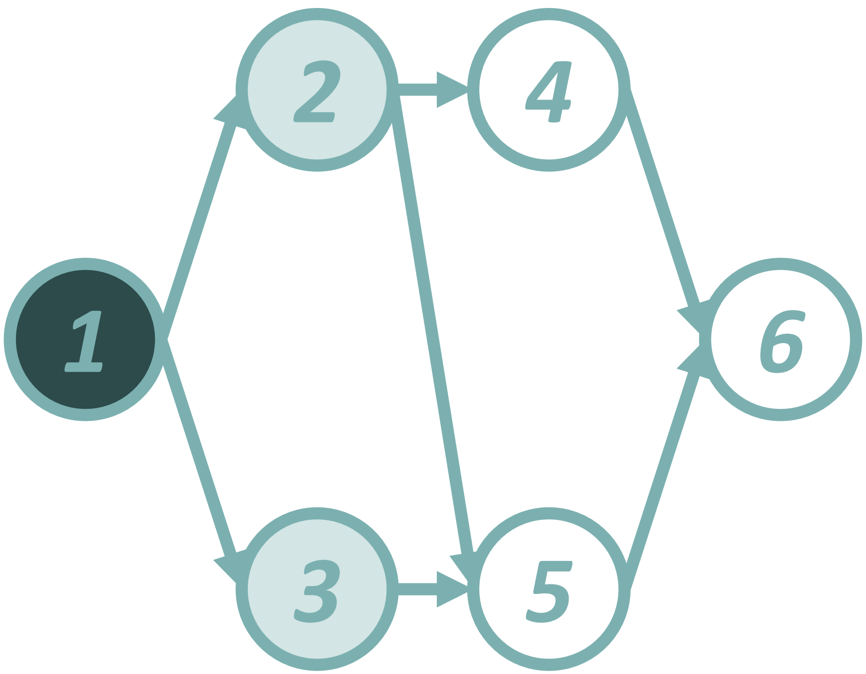
- What activities are ongoing? 2? 3? 2 and 3?
- Preemption is possible

Example: State space

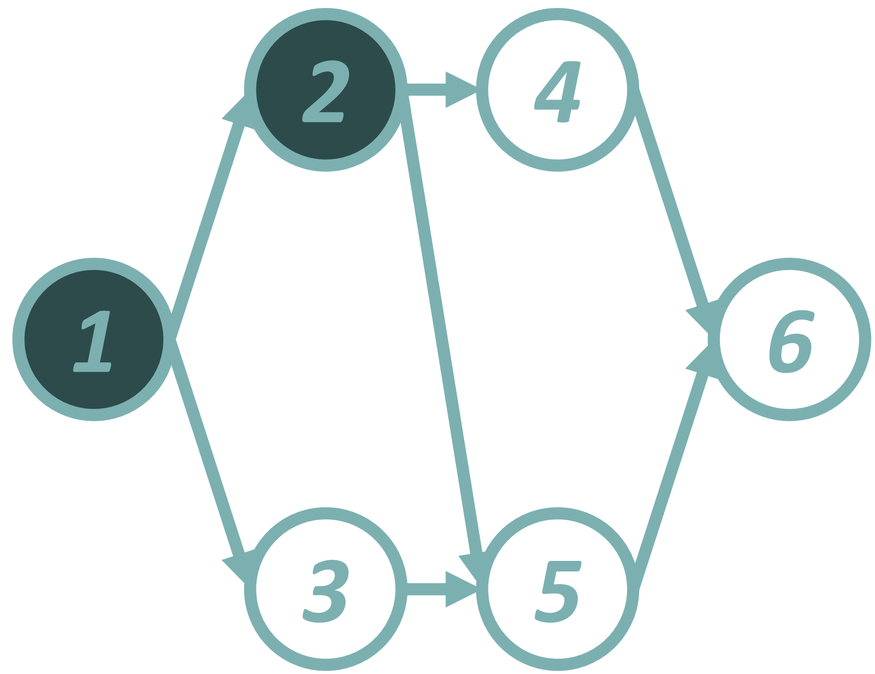


In this state, it is optimal if activities 2 & 3 are ongoing

Example: State space

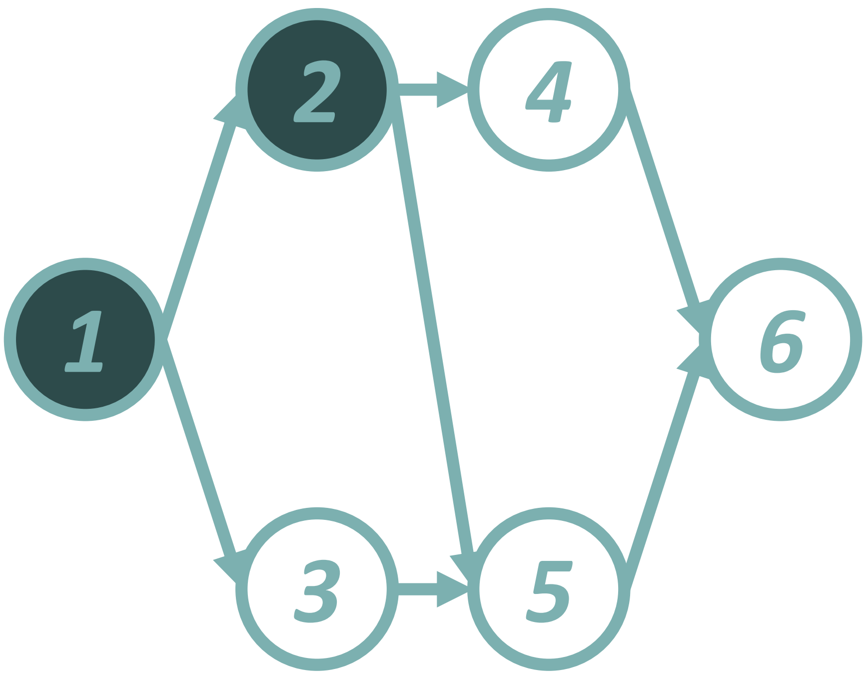


In this state, it is optimal if activities 2 & 3 are ongoing



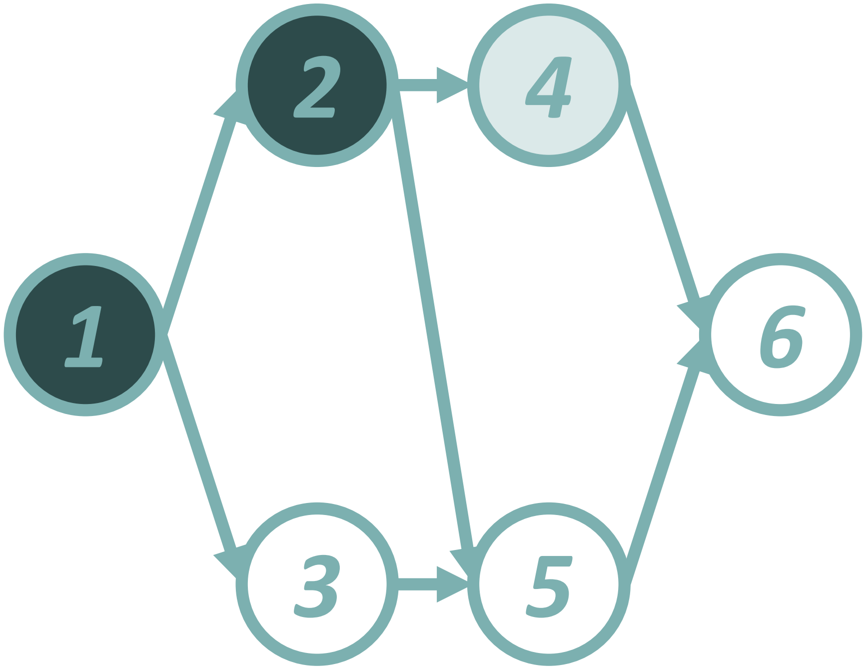
Activity 2 finishes ➔ we end up in state θ = *{1,1,0,0,0,0}*

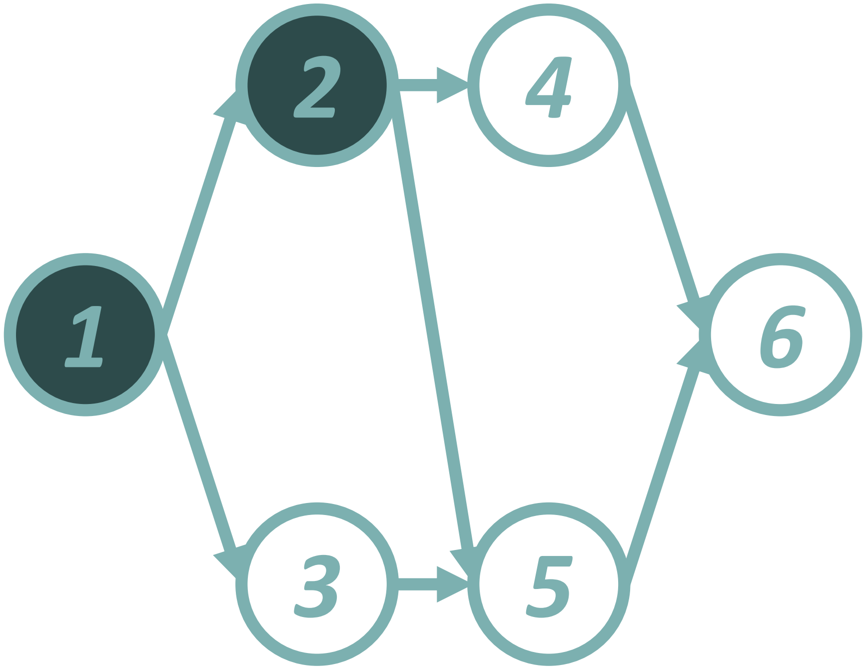
Example: State space



Activity 2 finishes ➔ we end up in state θ = *{1,1,0,0,0,0}*

Example: State space





Here, it is optimal if activity 4 is ongoing ➔activity 3 is preempted!

Activity 2 finishes ➔we end up in state θ = {1,1,0,0,0,0}

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**

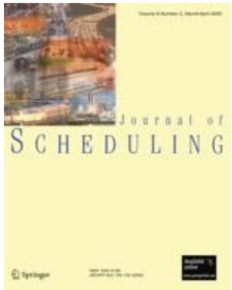
Creemers (2015)

- 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

SRCPSP

2015 (JOS) Instances Solved

OLD CTMC	
Instances solved (out of 480)	
J30	480
J60	303
J90	NA
J120	NA

SRCPSP

2015 (JOS) CPU Times

OLD CTMC	
Instances solved (out of 480)	
J30	480
J60	303
J90	NA
J120	NA

OLD CTMC	
Average CPU time (s)	
J30	0.48
J60	1591
J90	NA
J120	NA

SRCPSP

2015 (JOS) VS new CTMC

NEW CTMC	
Avg CPU time (s) for same inst.	
J30	0.02
J60	81.6
J90	NA
J120	NA

OLD CTMC	
Average CPU time (s)	
J30	0.48
J60	1591
J90	NA
J120	NA

SRCPSP

2015 (JOS) VS new CTMC

NEW CTMC	
Avg CPU time (s) for same inst.	
J30	0.02
J60	81.6
J90	NA
J120	NA

OLD CTMC	
Average CPU time (s)	
J30	0.48
J60	1591
J90	NA
J120	NA

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

SRCPSP

2015 (JOS) Memory Requirements

OLD CTMC	
Instances solved (out of 480)	
J30	480
J60	303
J90	NA
J120	NA

SRCPSP

2015 (JOS) Memory Requirements

OLD CTMC	
Instances solved (out of 480)	
J30	480
J60	303
J90	NA
J120	NA

OLD CTMC	
Average max # states (x1000)	
J30	176
J60	374499
J90	NA
J120	NA

SRCPSP

2015 (JOS) VS new CTMC

NEW CTMC	
Avg max # states (x1K) for = inst.	
J30	1.99
J60	508
J90	NA
J120	NA

OLD CTMC	
Average max # states (x1000)	
J30	176
J60	374499
J90	NA
J120	NA

SRCPSP

2015 (JOS) VS new CTMC

NEW CTMC	
Avg max # states (x1K) for = inst.	
J30	1.99
J60	508
J90	NA
J120	NA

OLD CTMC	
Average max # states (x1000)	
J30	176
J60	374499
J90	NA
J120	NA

On average, we reduce memory requirements by a factor of 733!

SRCPSP

New CTMC Instances Solved

NEW CTMC	
Instances solved (out of 480)	
J30	480
J60	480
J90	196
J120	10

SRCPSP

New CTMC Instances Solved

NEW CTMC	
Instances solved (out of 480)	
J30	480
J60	480
J90	196
J120	10

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

Creemers, Leus, & Lambrecht (2010)

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

OLD CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	29
n = 50	30	30	4
n = 60	30	30	0
n = 70	30	22	0

SNPV

2010 (ORL) CPU Times

OLD CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	29
n = 50	30	30	4
n = 60	30	30	0
n = 70	30	22	0

OLD CTMC			
Average CPU time (s)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	27
n = 40	0	7	2338
n = 50	0	100	52268
n = 60	1	2210	NA
n = 70	3	17496	NA

SNPV

2010 (ORL) VS new CTMC

NEW CTMC			
Average CPU time (s) for same instances			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	0
n = 40	0	0	7
n = 50	0	1	82
n = 60	0	6	NA
n = 70	0	34	NA

OLD CTMC			
Average CPU time (s)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	27
n = 40	0	7	2338
n = 50	0	100	52268
n = 60	1	2210	NA
n = 70	3	17496	NA

SNPV

2010 (ORL) VS new CTMC

NEW CTMC			
Average CPU time (s) for same instances			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	0
n = 40	0	0	7
n = 50	0	1	82
n = 60	0	6	NA
n = 70	0	34	NA

OLD CTMC			
Average CPU time (s)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	27
n = 40	0	7	2338
n = 50	0	100	52268
n = 60	1	2210	NA
n = 70	3	17496	NA

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

SNPV

2010 (ORL) Memory Requirements

OLD CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	29
n = 50	30	30	4
n = 60	30	30	0
n = 70	30	22	0

SNPV

2010 (ORL) Memory Requirements

OLD CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	29
n = 50	30	30	4
n = 60	30	30	0
n = 70	30	22	0

OLD CTMC			
Average max # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	1
n = 20	0	4	55
n = 30	2	49	1560
n = 40	8	534	47073
n = 50	27	4346	526020
n = 60	92	42279	NA
n = 70	287	216028	NA

SNPV

2010 (ORL) VS new CTMC

NEW CTMC			
Avg max # states (x1000) for same inst.			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	2
n = 30	0	2	17
n = 40	1	9	172
n = 50	2	40	1055
n = 60	4	175	NA
n = 70	8	593	NA

OLD CTMC			
Average max # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	1
n = 20	0	4	55
n = 30	2	49	1560
n = 40	8	534	47073
n = 50	27	4346	526020
n = 60	92	42279	NA
n = 70	287	216028	NA

SNPV

2010 (ORL) VS new CTMC

NEW CTMC			
Avg max # states (x1000) for same inst.			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	2
n = 30	0	2	17
n = 40	1	9	172
n = 50	2	40	1055
n = 60	4	175	NA
n = 70	8	593	NA

OLD CTMC			
Average max # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	1
n = 20	0	4	55
n = 30	2	49	1560
n = 40	8	534	47073
n = 50	27	4346	526020
n = 60	92	42279	NA
n = 70	287	216028	NA

On average, we reduce memory requirements by a factor of 403!

SNPV

New CTMC Instances Solved

NEW CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	30
n = 50	30	30	30
n = 60	30	30	30
n = 70	30	30	30

SNPV

New CTMC CPU Times

NEW CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	30
n = 50	30	30	30
n = 60	30	30	30
n = 70	30	30	30

NEW CTMC			
Average CPU time (s)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	0
n = 40	0	0	22
n = 50	0	1	476
n = 60	0	11	16869
n = 70	0	99	263012

SNPV

New CTMC CPU Times

NEW CTMC			
Instances solved (out of 30)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	30	30	30
n = 20	30	30	30
n = 30	30	30	30
n = 40	30	30	30
n = 50	30	30	30
n = 60	30	30	30
n = 70	30	30	30

NEW CTMC			
Average CPU time (s)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0	0	0
n = 20	0	0	0
n = 30	0	0	0
n = 40	0	0	22
n = 50	0	1	476
n = 60	0	11	16869
n = 70	0	99	263012

CPU times have become the new bottleneck

SNPV

To preempt or not to preempt?

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*)

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

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

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

Conclusion

- New CTMC that only keeps track of finished activities

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

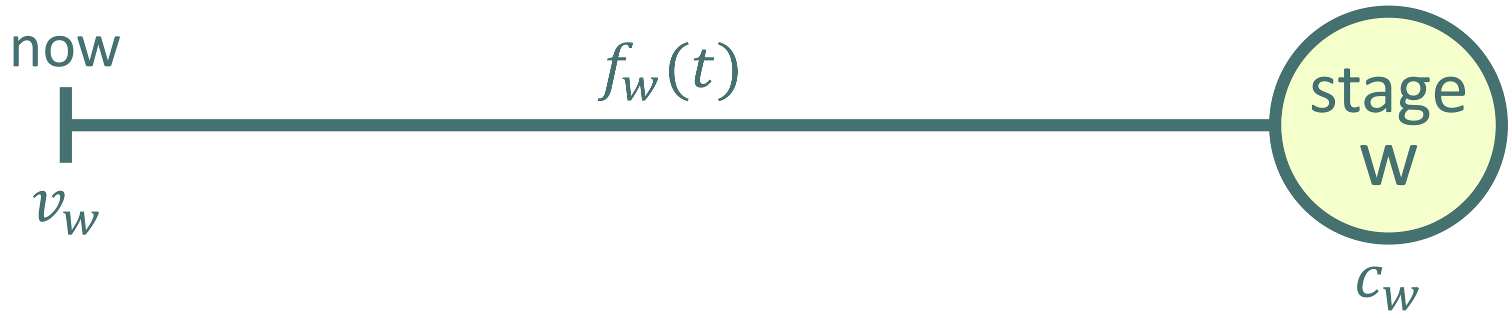
NPV of a single cash flow obtained after a single stage

- c_w = cash flow incurred at start of stage w

NPV of a single cash flow obtained after a single stage

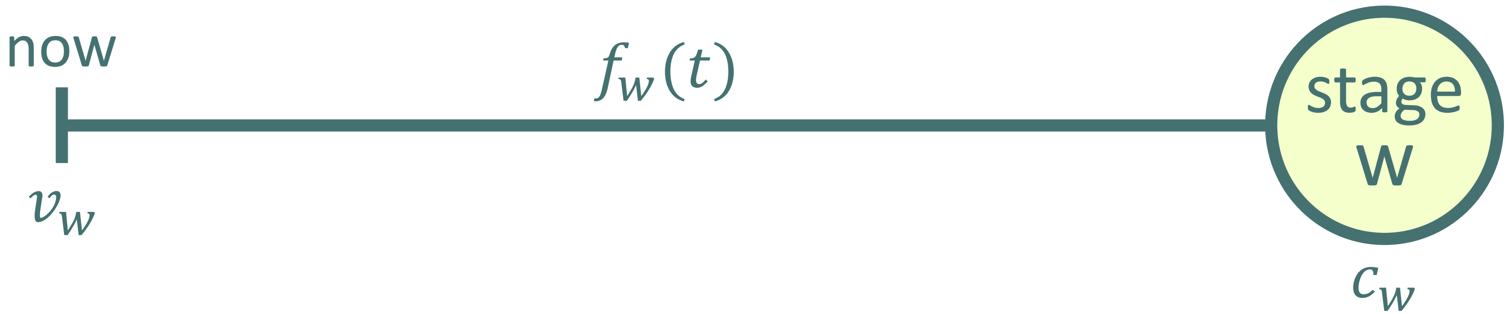
- c_w = cash flow incurred at start of stage w
- v_w = NPV of cash flow c_w

NPV of a single cash flow obtained after a single stage



- c_w = cash flow incurred at start of stage w
- v_w = NPV of cash flow c_w
- $f_w(t)$ = distribution of time until cash flow c_w is incurred

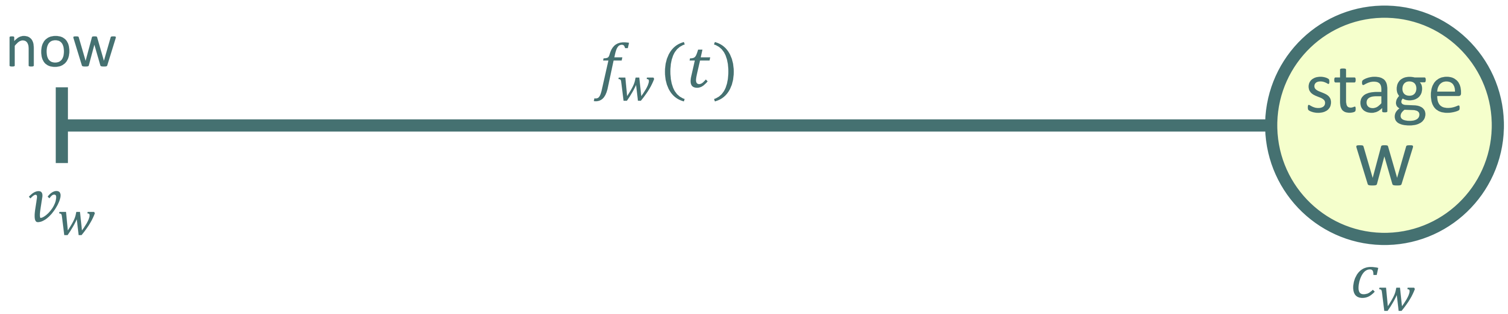
NPV of a single cash flow obtained after a single stage



$$v_w = c_w \int_0^\infty f_w(t)\, e^{-rt}\, dt$$

- c_w = cash flow incurred at start of stage w
- v_w = NPV of cash flow c_w
- $f_w(t)$ = distribution of time until cash flow c_w is incurred
- r = discount rate

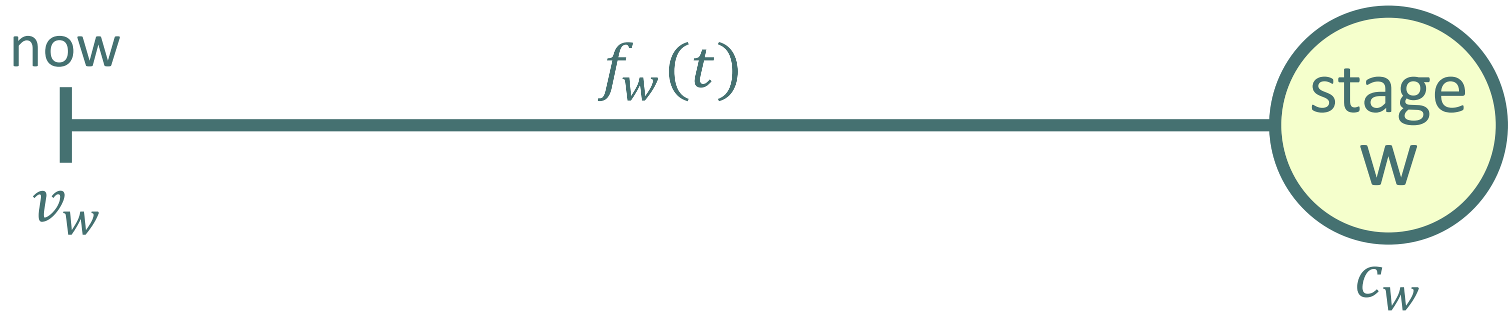
NPV of a single cash flow obtained after a single stage



$$v_w = c_w \int_0^\infty f_w(t)\, e^{-rt}\, dt \qquad v_w = c_w M_{f_w(t)}(-r)$$

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

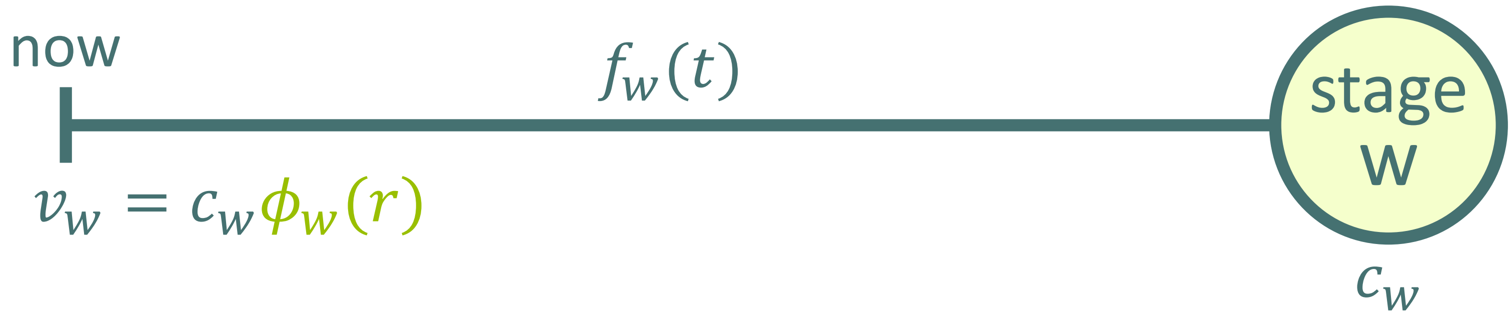
NPV of a single cash flow obtained after a single stage



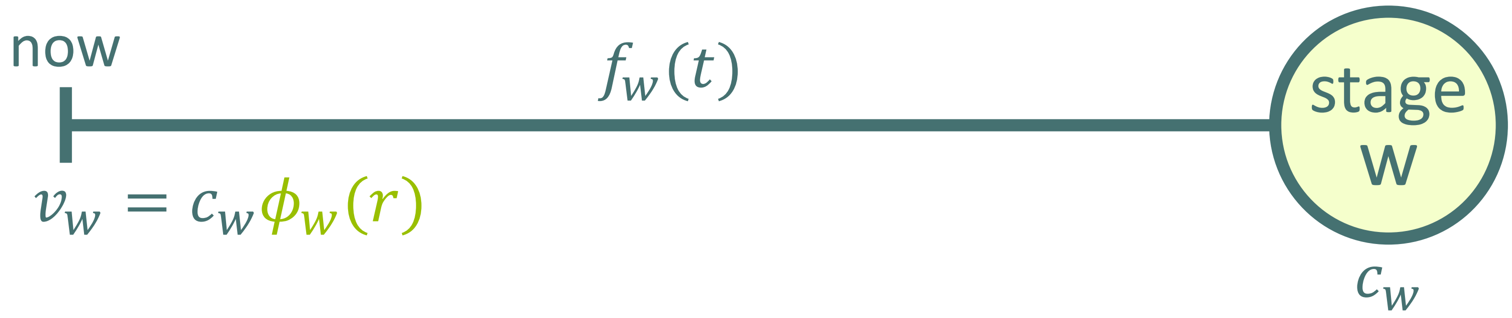
$$v_w = c_w \int_0^\infty f_w(t)\, e^{-rt}\, dt \qquad v_w = c_w M_{f_w(t)}(-r) \qquad v_w = c_w \phi_w(r)$$

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

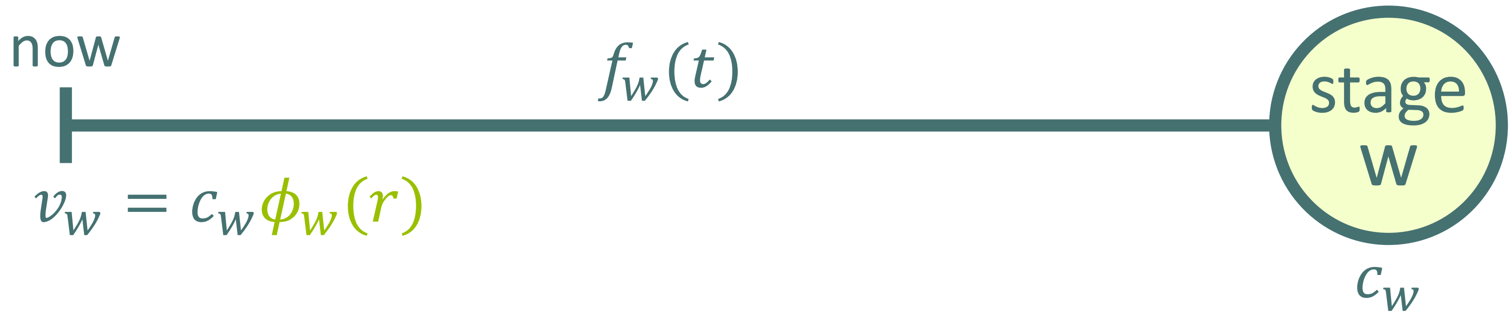


NPV of a single cash flow obtained after a single stage



- Using discount factor $\phi_w(r)$, we can obtain the moments of the NPV:
  - $\mu_w = c_w \phi_w(r)$
  - $\sigma_w^2 = c_w^2(\phi_w(2r) - \phi_w^2(r))$
  - $\gamma_w = c_w^3(\phi_w(3r) - 3\phi_w(2r)\phi_w(r) + 2\phi_w^3(r))\, \sigma_w^{-3}$
  - $\theta_w = c_w^4(\phi_w(4r) - 4\phi_w(3r)\phi_w(r) + 6\phi_w(2r)\phi_w^2(r) - 3\phi_w^4(r))\, \sigma_w^{-4}$

NPV of a single cash flow obtained after a single stage



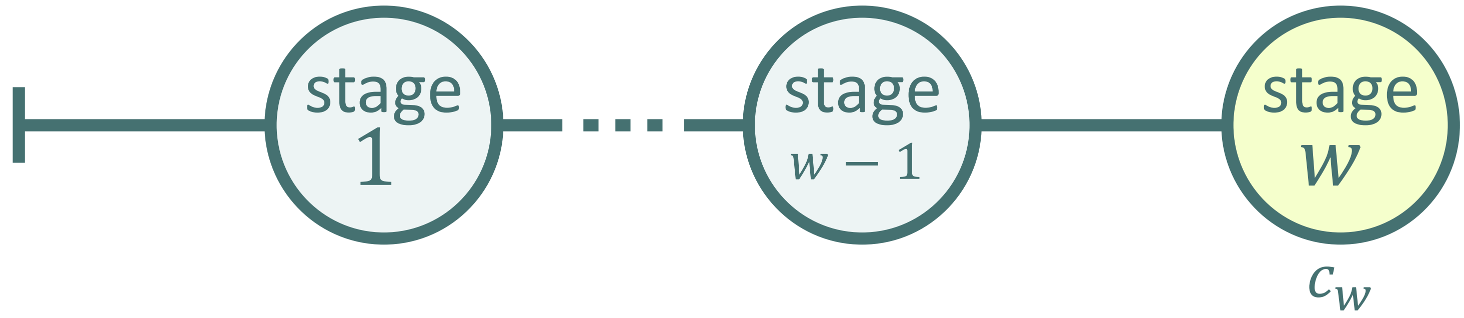
- Using discount factor $\phi_w(r)$, we can obtain the moments of the NPV:
  - $\mu_w = c_w \phi_w(r)$
  - $\sigma_w^2 = c_w^2 (\phi_w(2r) - \phi_w^2(r))$
  - $\gamma_w = c_w^3 (\phi_w(3r) - 3\phi_w(2r)\phi_w(r) + 2\phi_w^3(r))\, \sigma_w^{-3}$
  - $\theta_w = c_w^4 (\phi_w(4r) - 4\phi_w(3r)\phi_w(r) + 6\phi_w(2r)\phi_w^2(r) - 3\phi_w^4(r))\, \sigma_w^{-4}$
- The CDF & PDF of the NPV of c_w are:
  - $G_w(v) = 1 - F_w\left(\ln\left(\frac{c_w}{v}\right) r^{-1}\right)$
  - $g_w(v) = \frac{f_w\left(\ln\left(\frac{c_w}{v}\right) r^{-1}\right)}{|r| v}$

Agenda

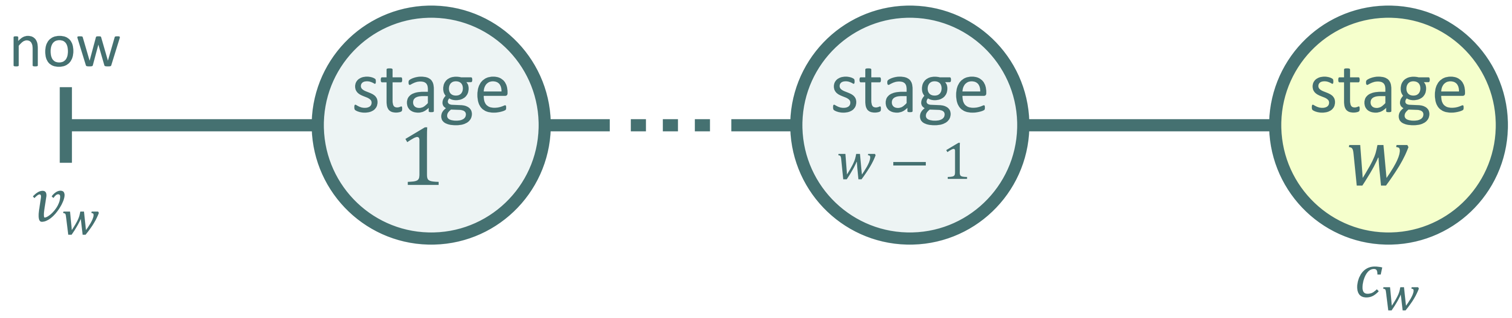
- Introduction
- Serial projects:
  - Single cash flow after a single stage
  - Single cash flow after multiple stages
  - NPV of a serial project
  - Optimal sequence of stages
- General projects
- Conclusions

NPV of a single cash flow obtained after multiple stages

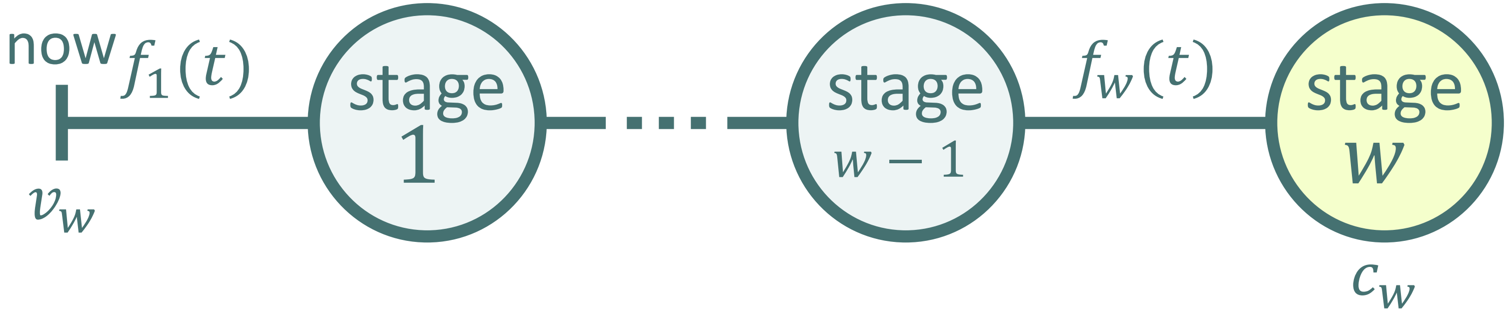
NPV of a single cash flow obtained after multiple stages



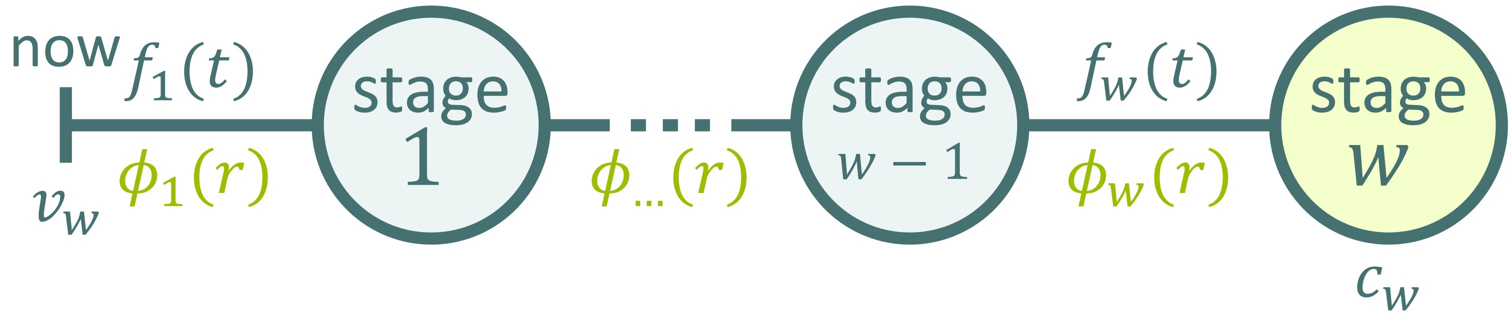
NPV of a single cash flow obtained after multiple stages



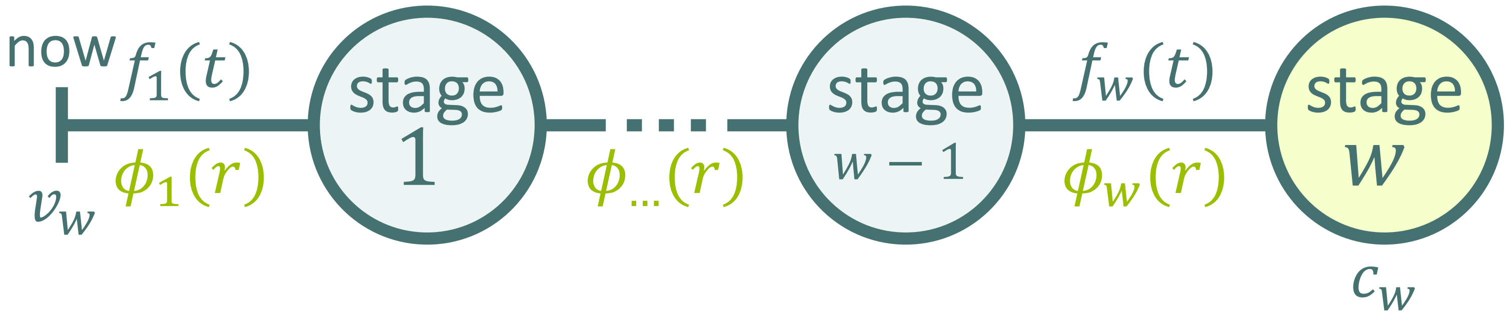
NPV of a single cash flow obtained after multiple stages



NPV of a single cash flow obtained after multiple stages

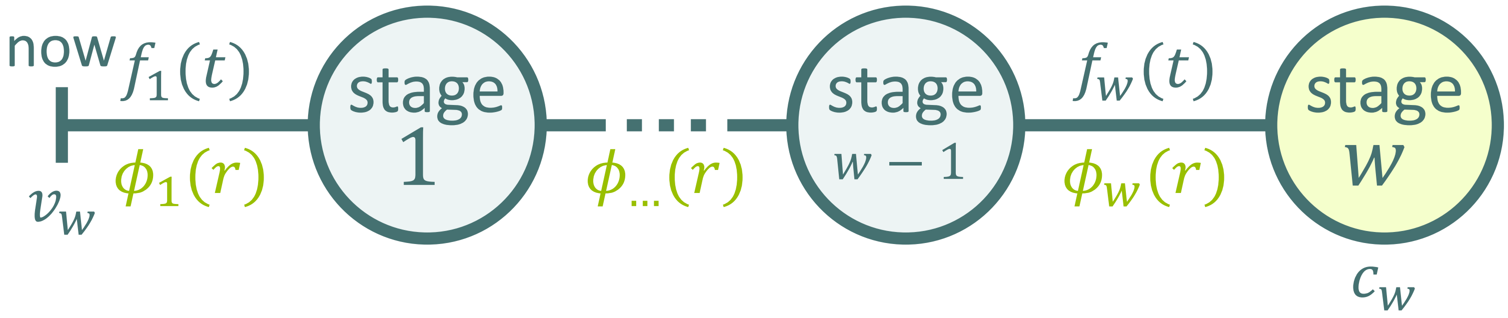


NPV of a single cash flow obtained after multiple stages



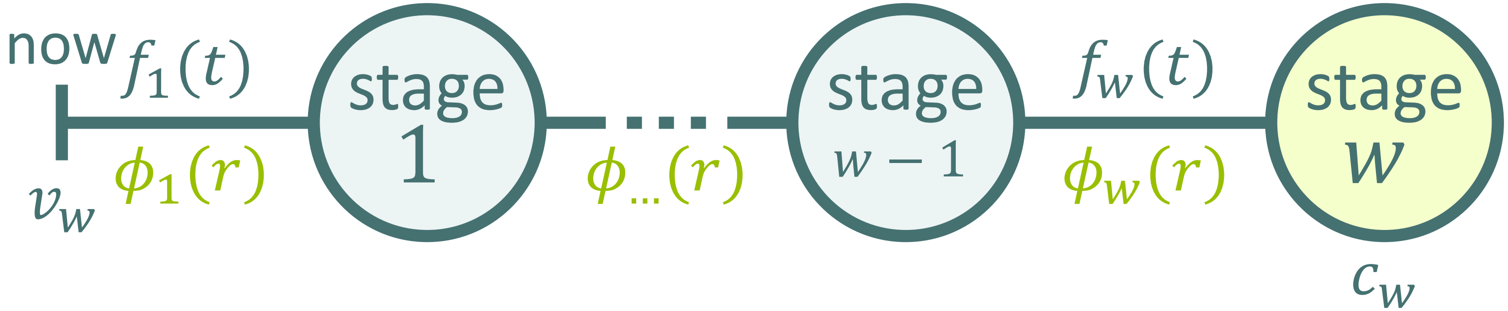
$$v_w = c_w \phi_1(r) \dots \phi_w(r)$$

NPV of a single cash flow obtained after multiple stages



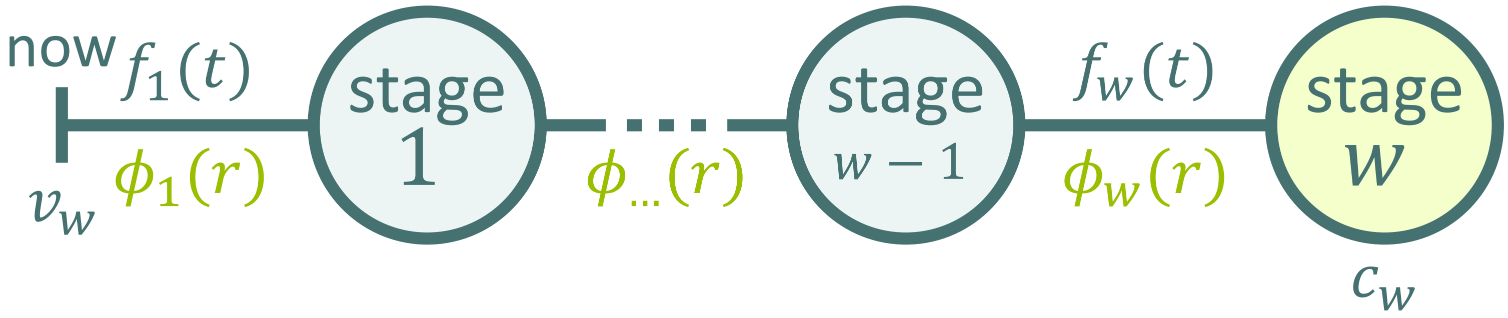
$v_w = c_w \phi_1(r) \dots \phi_w(r)$ $\quad$ $v_w = c_w \prod_{i=1}^{w} \phi_i(r)$

NPV of a single cash flow obtained after multiple stages



$$v_w = c_w \phi_1(r) \ldots \phi_w(r) \qquad v_w = c_w \prod_{i=1}^{w} \phi_i(r) \qquad v_w = c_w \phi_{1,w}(r)$$

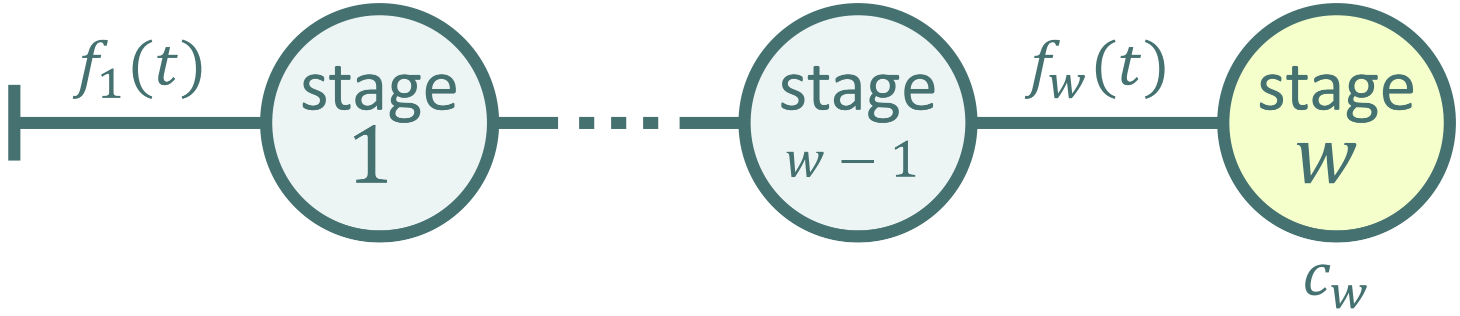
NPV of a single cash flow obtained after multiple stages



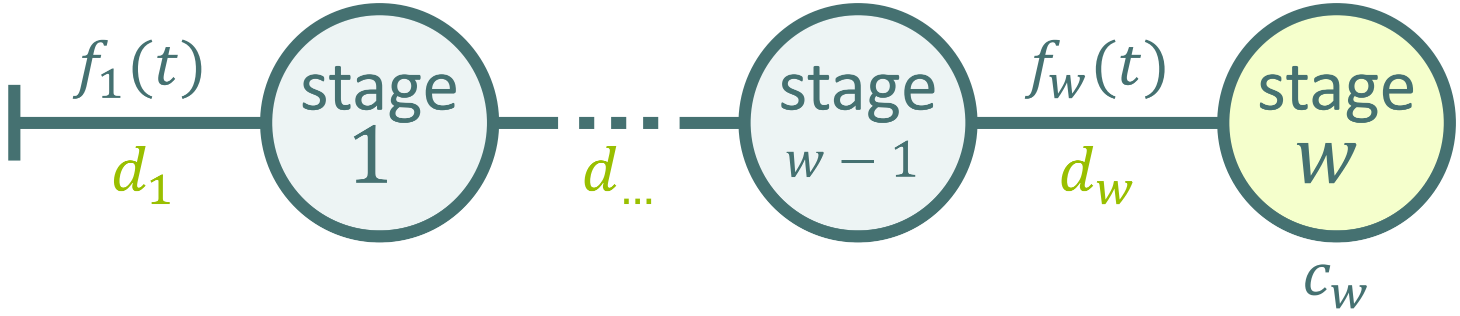
$$v_w = c_w \phi_1(r) \dots \phi_w(r) \qquad v_w = c_w \prod_{i=1}^{w} \phi_i(r) \qquad v_w = c_w \phi_{1,w}(r)$$

- We can obtain the moments of the NPV of cash flow c_w:
  - $\mu_w = c_w \phi_{1,w}(r)$
  - $\sigma_w^2 = c_w^2(\phi_{1,w}(2r) - \phi_{1,w}^2(r))$
  - …

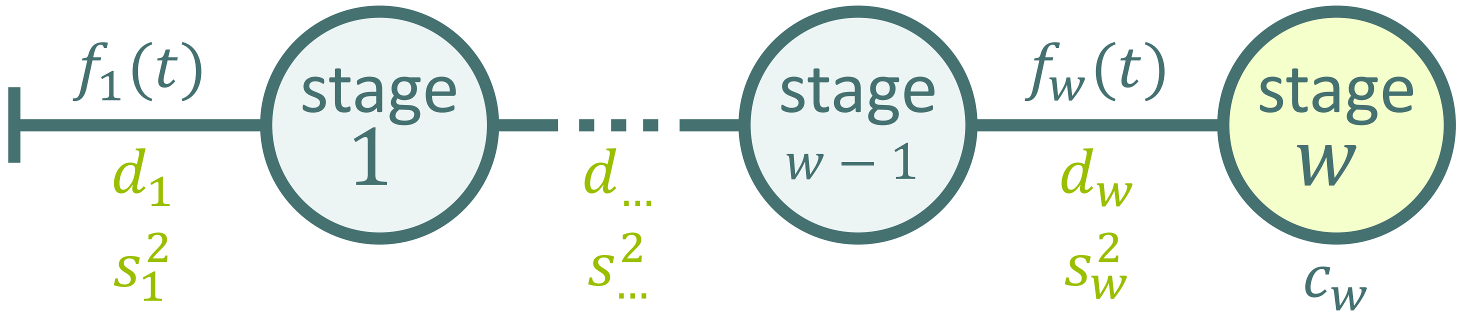
NPV of a single cash flow obtained after multiple stages



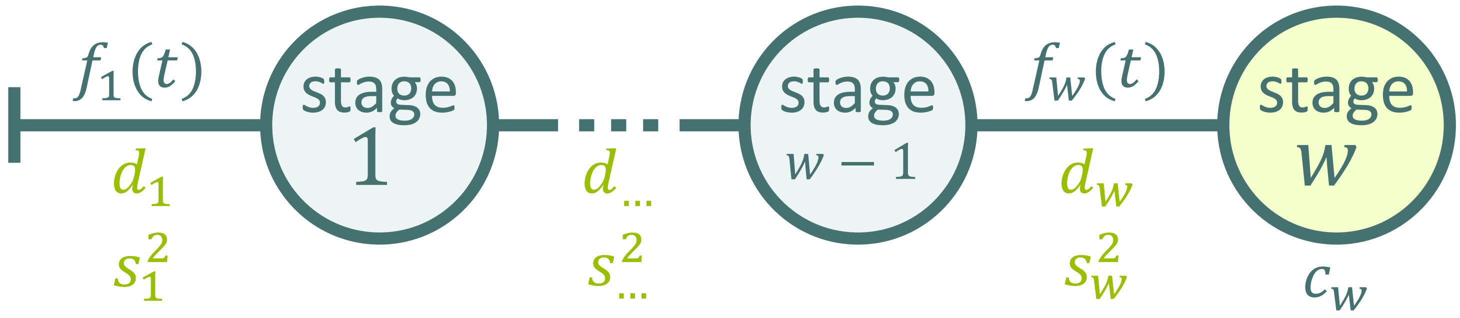
NPV of a single cash flow obtained after multiple stages



NPV of a single cash flow obtained after multiple stages

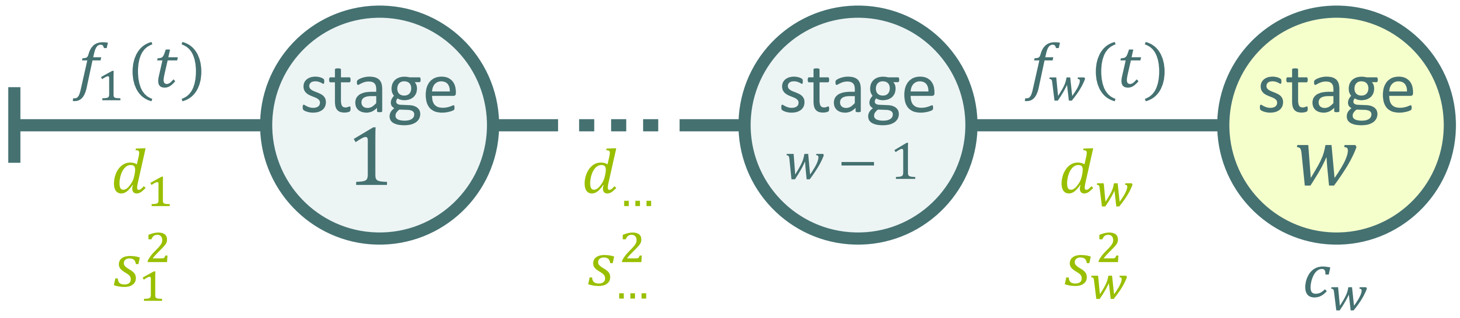


NPV of a single cash flow obtained after multiple stages



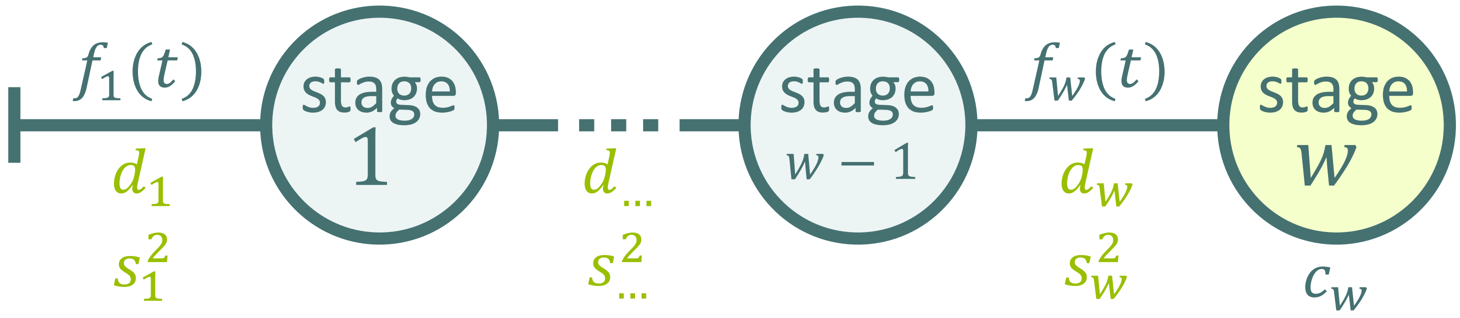
- The mean and variance of the distribution of time until cash flow c_w is incurred is:
  - $d_{1,w} = \sum_{i=1}^{w} d_i$
  - $s_{1,w}^2 = \sum_{i=1}^{w} s_i^2$

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 $s_{1,w}^2$

NPV of a single cash flow obtained after multiple stages



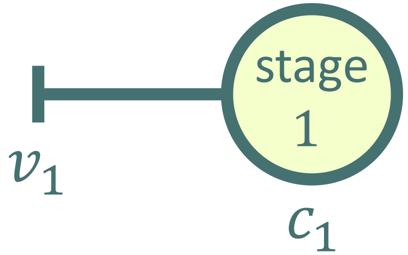
- The mean and variance of the distribution of time until cash flow c_w is incurred is:
  - $d_{1,w} = \sum_{i=1}^{w} d_i$
  - $s_{1,w}^2 = \sum_{i=1}^{w} s_i^2$
- If stage w is preceded by a sufficient number of stages, $f_{1,w}(t)$ is normally distributed with mean $d_{1,w}$ and variance $s_{1,w}^2$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

Agenda

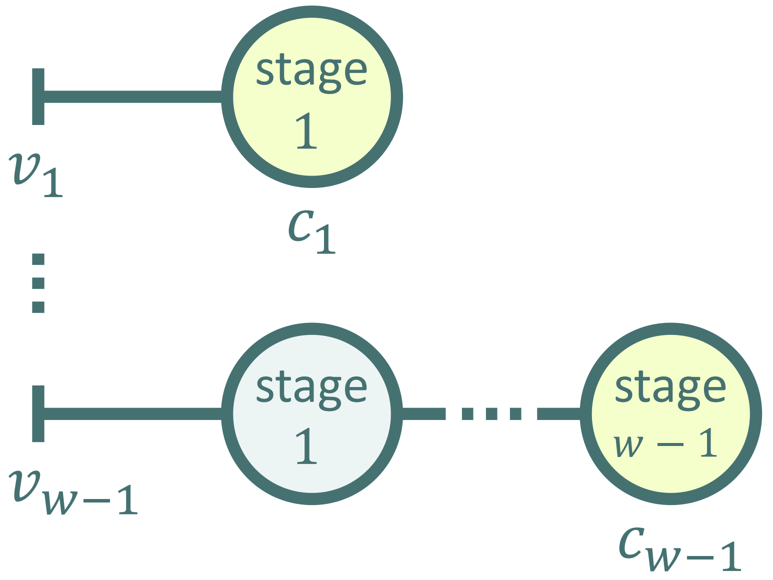
- Introduction
- Serial projects:
  - Single cash flow after a single stage
  - Single cash flow after multiple stages
  - NPV of a serial project
  - Optimal sequence of stages
- General projects
- Conclusions

NPV of a serial project

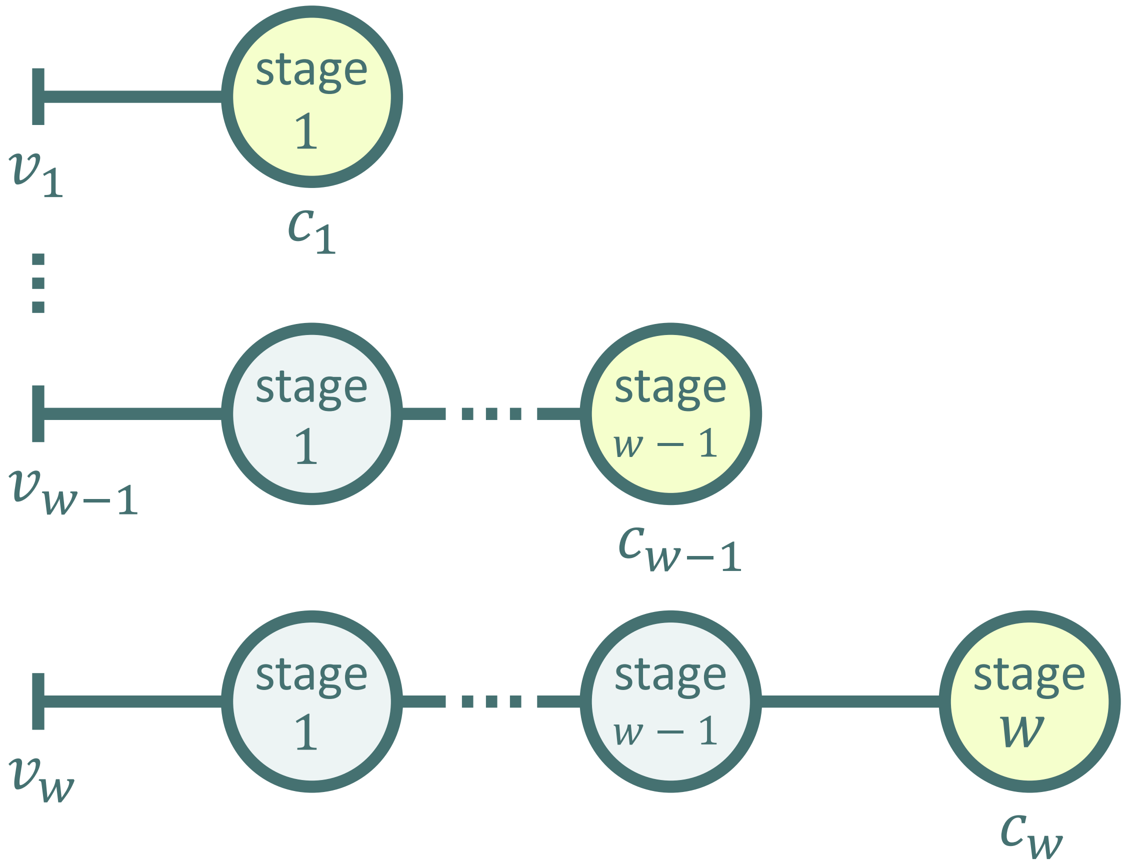
NPV of a serial project



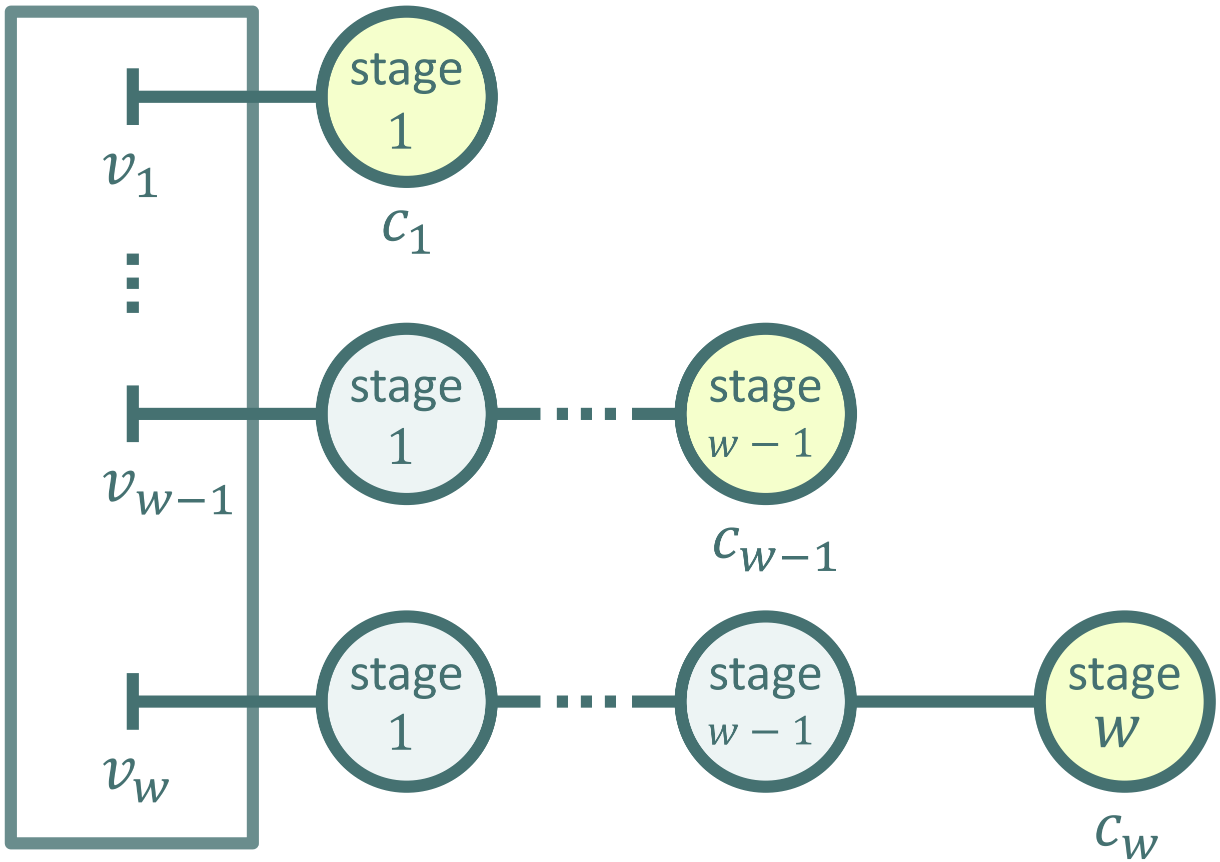
NPV of a serial project



NPV of a serial project



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:

Mean μ
$\mu_w = c_w a_1$

Covariance matrix $\boldsymbol{\Sigma}_c$
$\boldsymbol{\Sigma}_c(w,w) = \sigma_w^2 = c_w^2(a_2 - a^2)$
$\boldsymbol{\Sigma}_c(w,x) = c_w c_x b_1 \left(a_2 - a^2\right) = c_w^{-1} c_x b_1 \boldsymbol{\Sigma}_c(w,w)$

Central coskewness matrix $\boldsymbol{\Gamma}_c$
$\boldsymbol{\Gamma}_c(w,w,w) = \gamma_w \sigma_w^3 = c_w^3 \left(a_3 - 3a_2 a_1 + 2a^3\right)$
$\boldsymbol{\Gamma}_c(w,w,x) = c_w^{-1} c_x b_1 \boldsymbol{\Gamma}_c(w,w,w)$
$\boldsymbol{\Gamma}_c(w,x,x) = c_w c_x^2 \left(a_3 b_2 - a_2 a_1 \left(2b^2 + b_2\right) + 2a^3 b^2\right)$
$\boldsymbol{\Gamma}_c(w,x,y) = c_x^{-1} c_y h_1 \boldsymbol{\Gamma}_c(w,x,x)$

Central cokurtosis matrix $\boldsymbol{\Theta}_c$
$\boldsymbol{\Theta}_c(w,w,w,w) = \theta_w \sigma_w^4 = c_w^4 \left(a_4 - 4a_3 a_1 + 6a_2 a^2 - 3a^4\right)$
$\boldsymbol{\Theta}_c(w,w,w,x) = c_w^{-1} c_x b_1 \boldsymbol{\Theta}_c(w,w,w,w)$
$\boldsymbol{\Theta}_c(w,w,x,x) = c_w^2 c_x^2 \left(a_4 b_2 - 2a_3 a_1 \left(b_2 + b^2\right) + a_2 a^2 \left(b_2 + 5b^2\right) - 3a^4 b^2\right)$
$\boldsymbol{\Theta}_c(w,x,x,x) = c_w c_x^3 \left(a_4 b_3 - a_3 a_1 \left(b_3 + 3b_2 b_1\right) + 3a_2 a^2 \left(b_2 b_1 + b^3\right) - 3a^4 b^3\right)$
$\boldsymbol{\Theta}_c(w,w,x,y) = c_x^{-1} c_y h_1 \boldsymbol{\Theta}_c(w,w,x,x)$
$\boldsymbol{\Theta}_c(w,x,x,y) = c_x^{-1} c_y h_1 \boldsymbol{\Theta}_c(w,x,x,x)$
$\boldsymbol{\Theta}_c(w,x,y,y) = c_w c_x c_y^2 \left(\left(a_4 - a_3 a_1\right) b_3 h_2 - \left(h_2 + 2h^2\right) \left(\left(a_3 a_1 - a_2 a^2\right) b_2 b_1\right) + \left(a_2 a^2 - a^4\right) 3b^3 h^2\right)$
$\boldsymbol{\Theta}_c(w,x,y,z) = c_y^{-1} c_z o_1(r) \boldsymbol{\Theta}_c(w,x,y,y)$

$a_i = \phi_{1,w-1}(ir)$	$b_i = \phi_{w,x-1}(ir)$	$h_i = \phi_{x,y-1}(ir)$	$o_i = \phi_{y,z-1}(ir)$
$a^i = \phi^i_{1,w-1}(r)$	$b^i = \phi^i_{w,x-1}(r)$	$h^i = \phi^i_{x,y-1}(r)$	

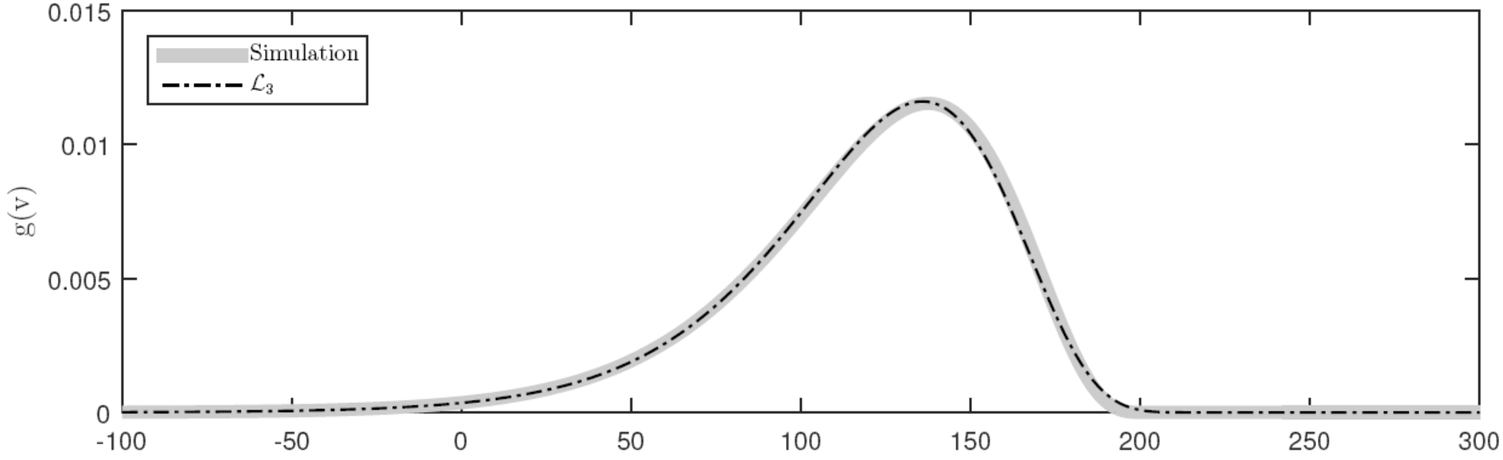
NPV of a serial project

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

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

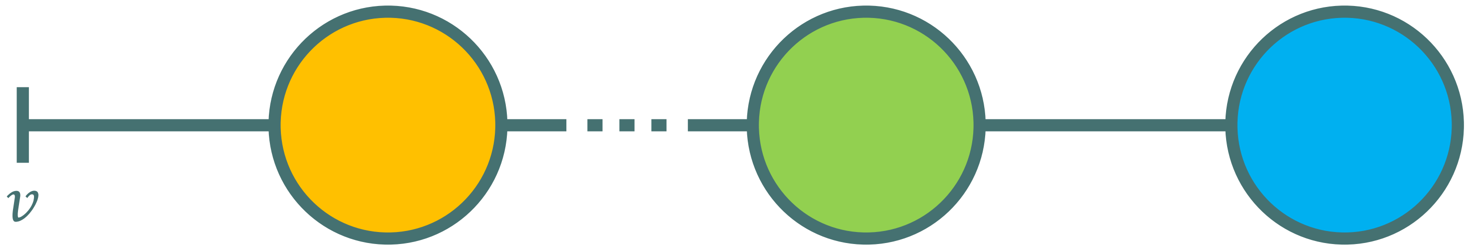
The example below illustrates the accuracy of the three-parameter lognormal distribution ($\mathcal{L}_3$):



Agenda

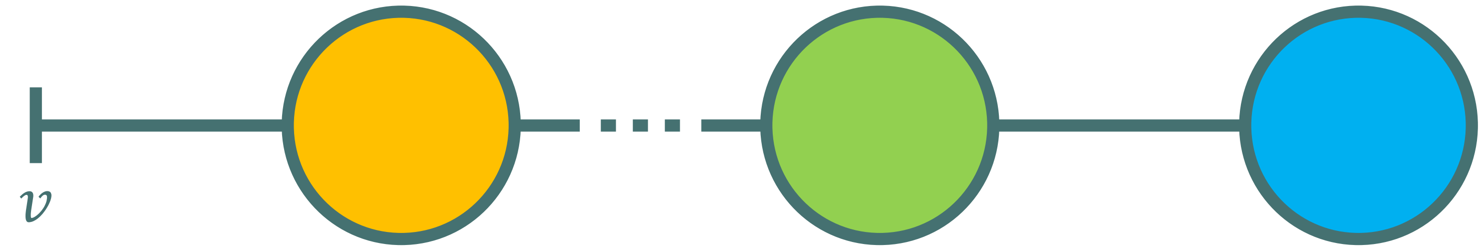
- Introduction
- Serial projects:
  - Single cash flow after a single stage
  - Single cash flow after multiple stages
  - NPV of a serial project
  - Optimal sequence of stages
- General projects
- Conclusions

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

Optimal sequence of stages

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

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

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

Optimal sequence of stages

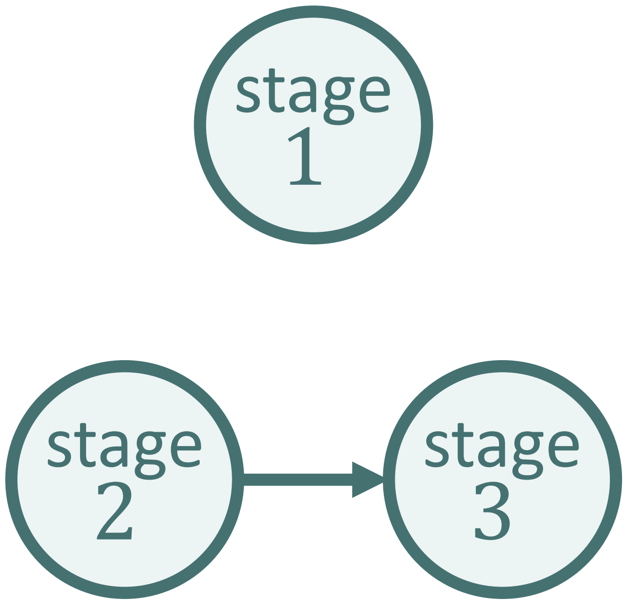
- The problem to find the optimal sequence of stages is equivalent to the Least Cost Fault Detection Problem (LCFDP)
- The LCFDP minimizes the cost of the sequential diagnosis of a number of system components
- In the absence of precedence relations, the optimal sequence can be found in polynomial time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

Agenda

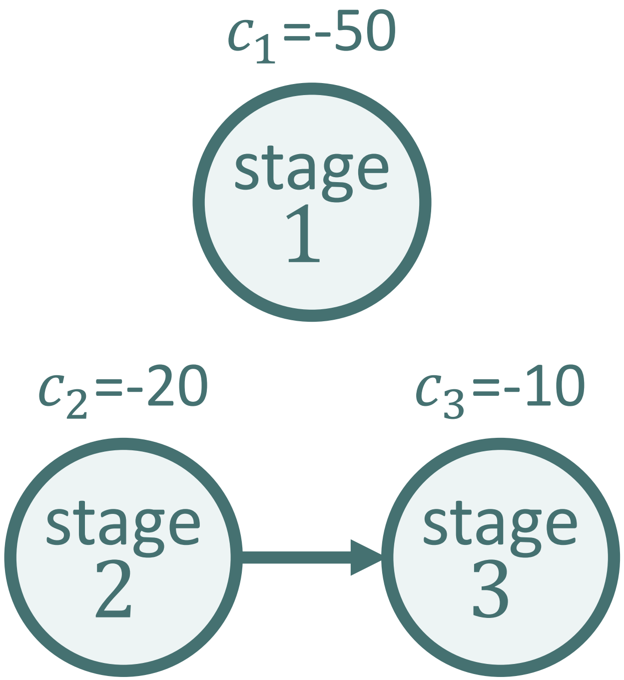
- Introduction
- Serial projects:
  - Single cash flow after a single stage
  - Single cash flow after multiple stages
  - NPV of a serial project
  - Optimal sequence of stages
- General projects
- Conclusions

NPV of a general project

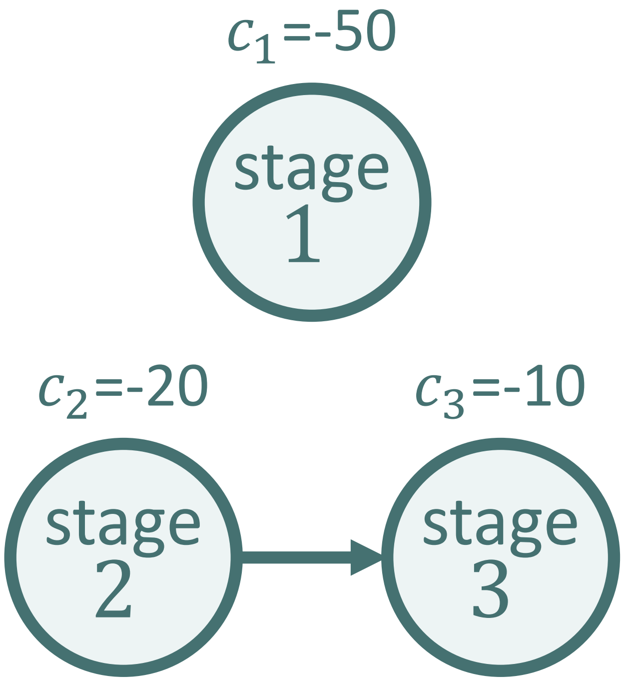
NPV of a general project
Scheduling policies



NPV of a general project
Scheduling policies



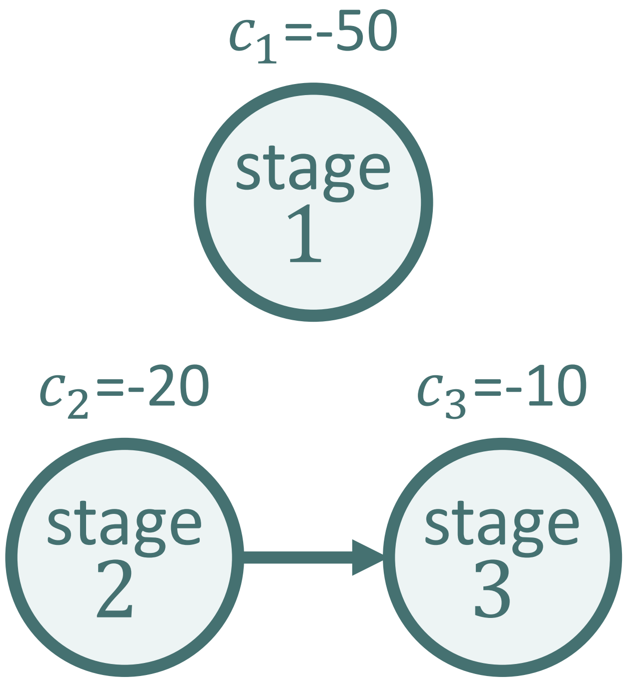
NPV of a general project
Scheduling policies



$f_1(t) \sim Exp(1)$

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

NPV of a general project
Scheduling policies

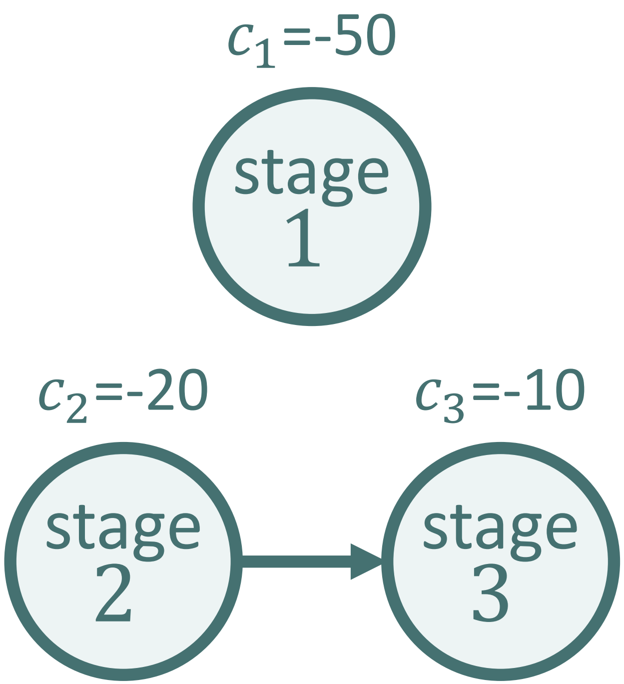


$f_1(t) \sim Exp(1)$

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

$p = 200$

NPV of a general project
Scheduling policies

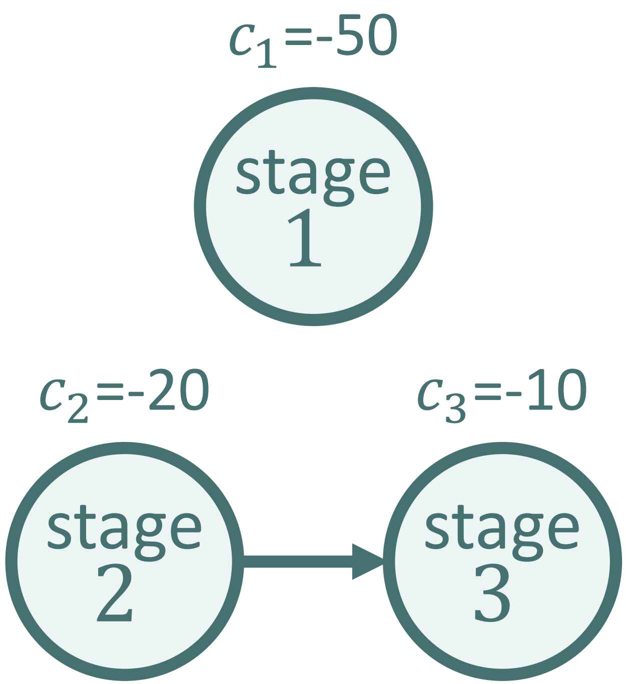


$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

NPV of a general project
Scheduling policies



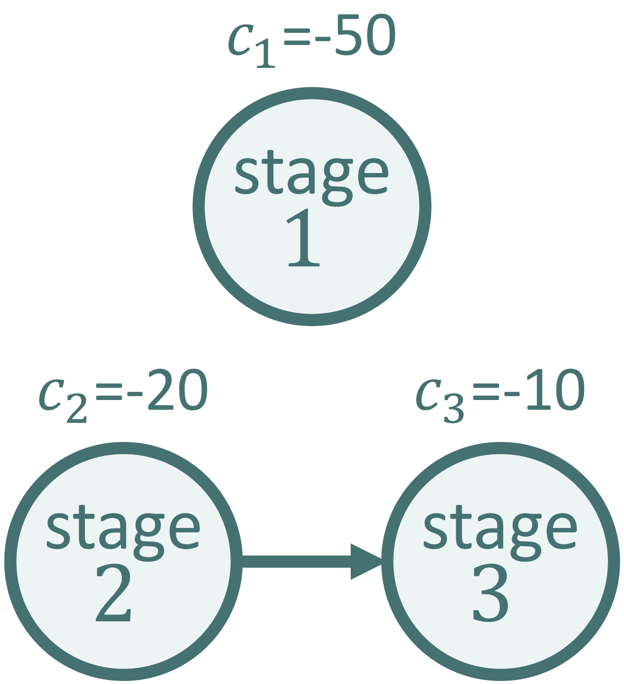
$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

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

NPV of a general project
Scheduling policies



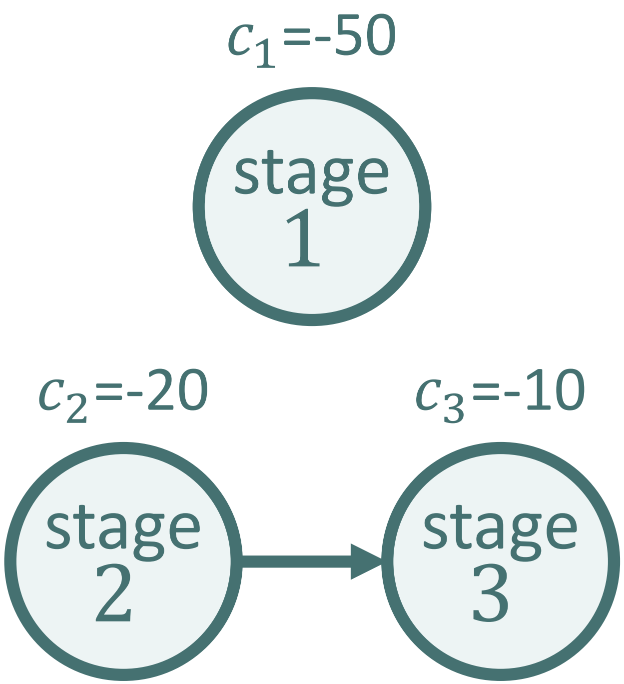
$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

- 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.

NPV of a general project
Scheduling policies



$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

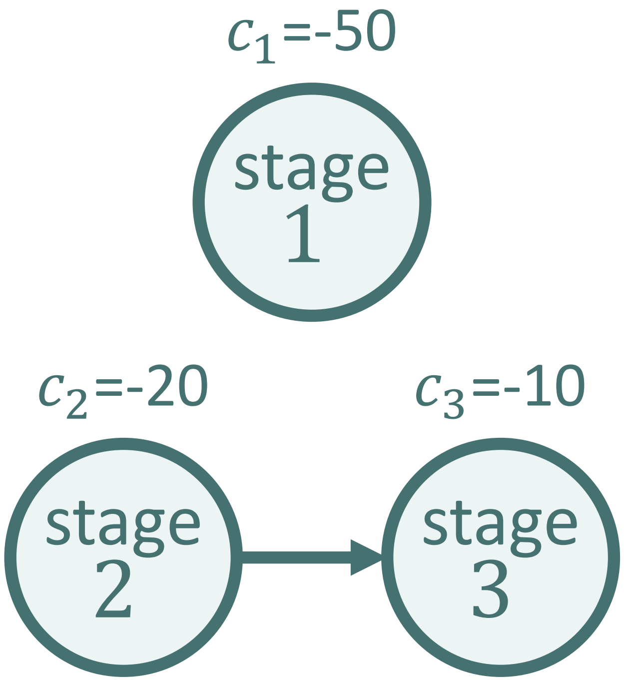
- Serial policies:
  - 1-2-3
  - 1-3-2
  - 2-1-3
  - 2-3-1
  - 3-1-2
  - 3-2-1
- Early-Start (ES) policy: Start 1 & 2. Start 3 upon completion of 2.

…

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

NPV of a general project

Early-Start policy



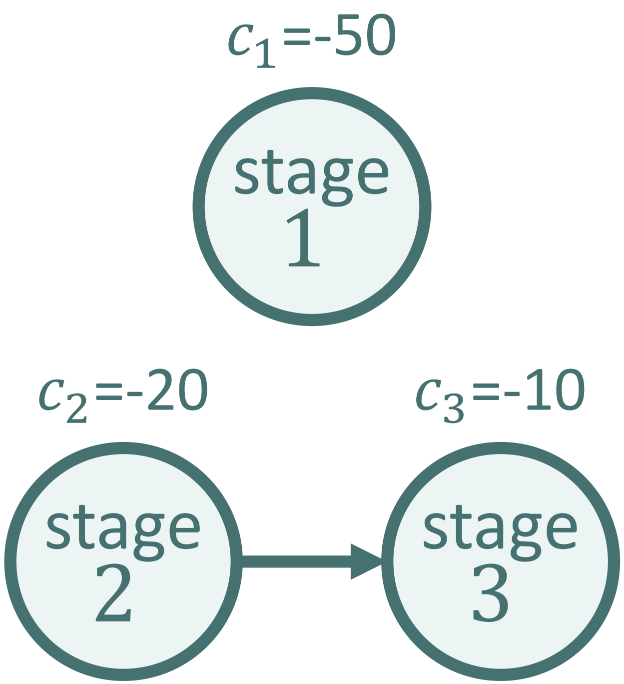
$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

NPV of a general project

Early-Start policy



$$f_1(t) \sim Exp(1)$$

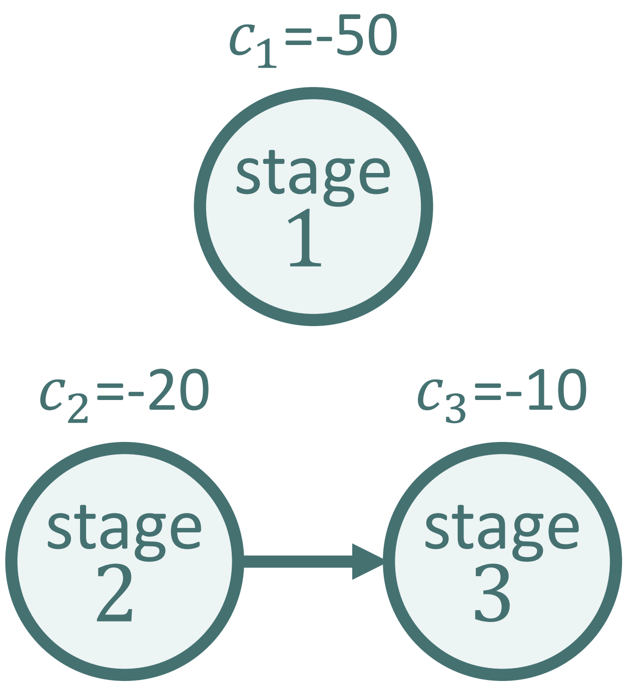
$$f_{2,3}(t) \sim Exp(0.5)$$

$$p = 200 \quad r = 0.1$$

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

NPV of a general project

Early-Start policy



$f_1(t) \sim Exp(1)$

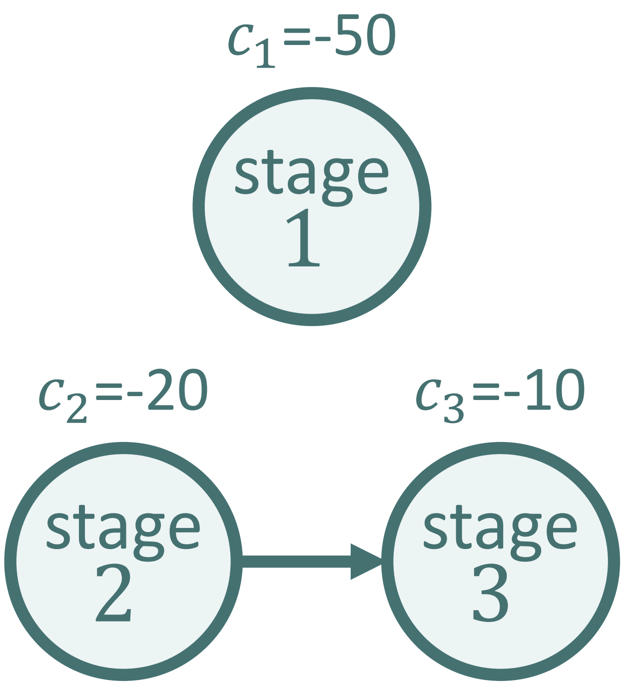
$f_{2,3}(t) \sim Exp(0.5)$

$p = 200 \quad r = 0.1$

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

NPV of a general project

Early-Start policy



$f_1(t) \sim Exp(1)$

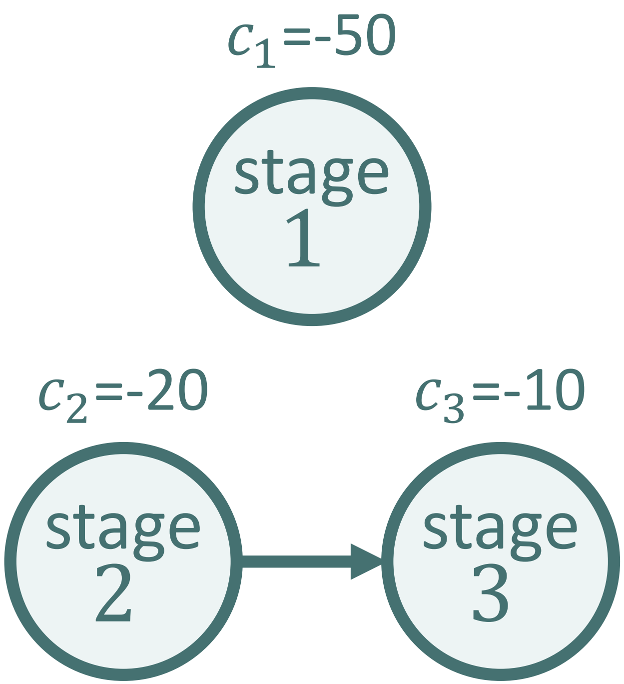
$f_{2,3}(t) \sim Exp(0.5)$

$p = 200 \quad r = 0.1$

- 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

NPV of a general project

Early-Start policy



$f_1(t) \sim Exp(1)$

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

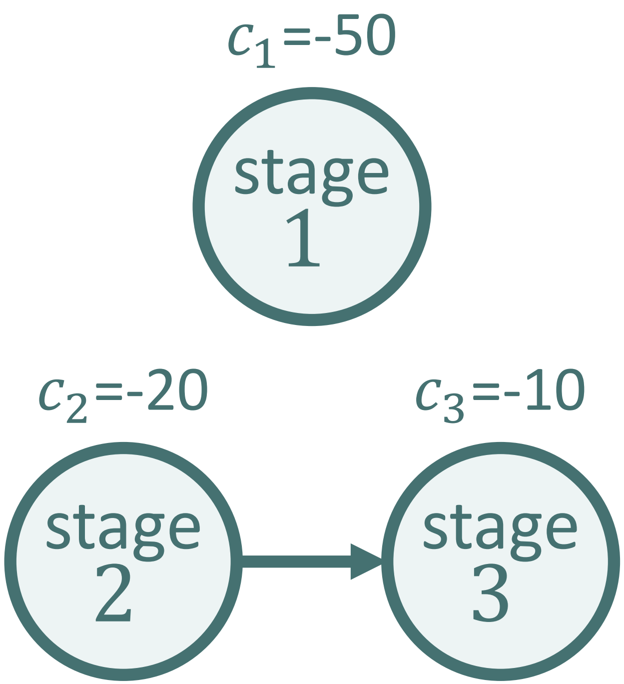
$p = 200 \quad r = 0.1$

- 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!

NPV of a general project

Optimal policy



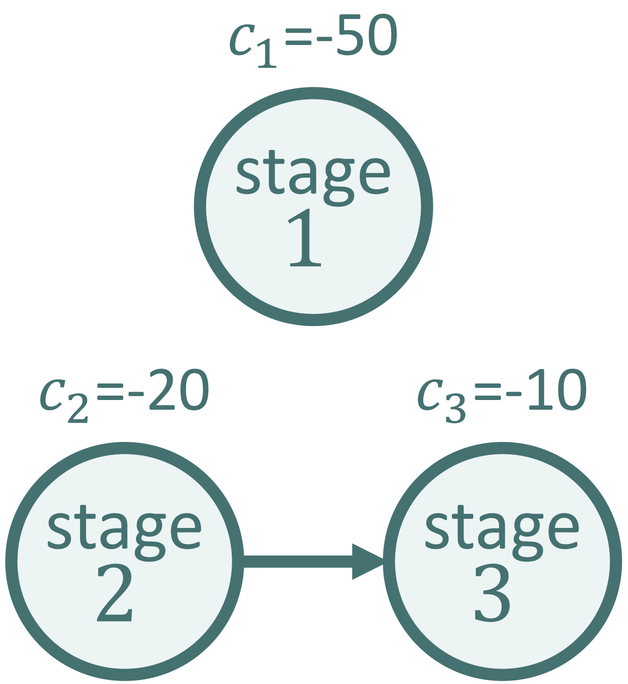
$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

NPV of a general project

Optimal policy



$f_1(t) \sim Exp(1)$

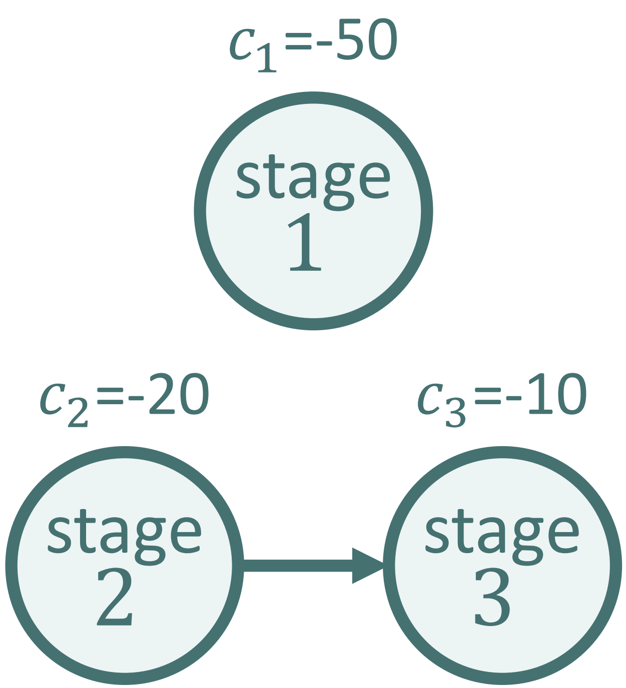
$f_{2,3}(t) \sim Exp(0.5)$

$p = 200 \quad r = 0.1$

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

NPV of a general project

Optimal policy



$f_1(t) \sim Exp(1)$

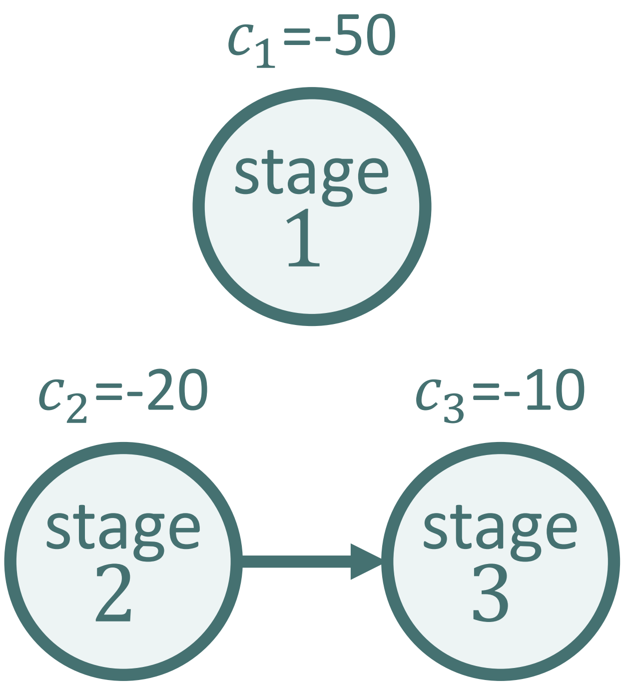
$f_{2,3}(t) \sim Exp(0.5)$

$p = 200 \quad r = 0.1$

- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel
- What discount factor do we use?
  - $\phi_2(r)\, \phi_1(r)$
  - $\phi_2(r)\, \phi_3(r)$

NPV of a general project

Optimal policy



$f_1(t) \sim Exp(1)$

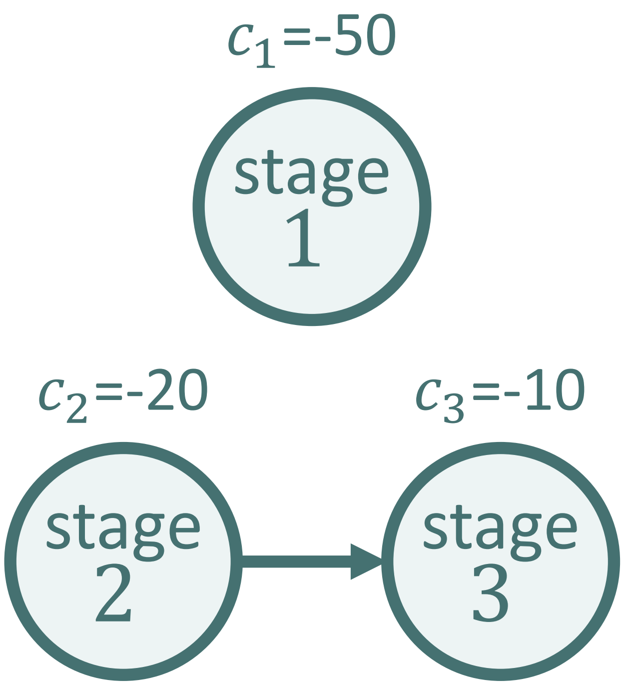
$f_{2,3}(t) \sim Exp(0.5)$

$p = 200 \quad r = 0.1$

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

NPV of a general project

Optimal policy



$f_1(t) \sim Exp(1)$

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

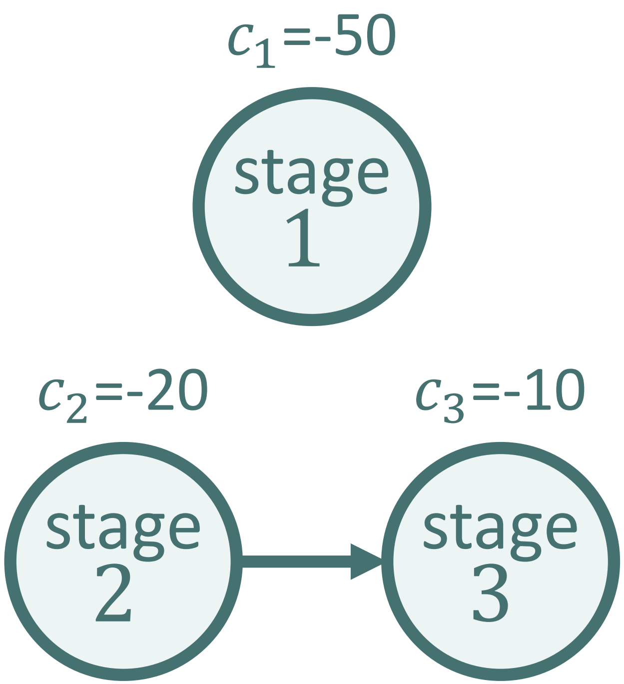
$p = 200 \quad r = 0.1$

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

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

NPV of a general project

Optimal policy



$f_1(t) \sim Exp(1)$

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

$p = 200 \quad r = 0.1$

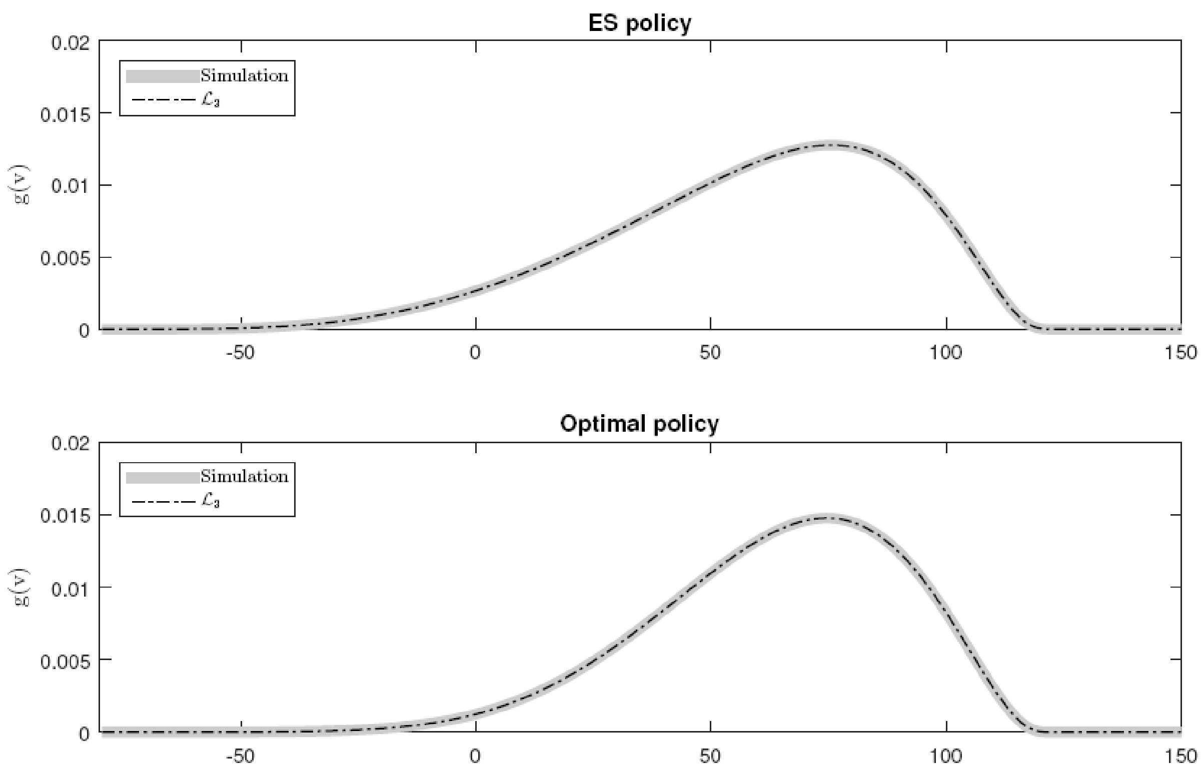
- Payoff is obtained after stage 2 & after stages 1 & 3 that are executed in parallel
- What discount factor do we use?
  - $\phi_2(r)\, \phi_1(r)$
  - $\phi_2(r)\, \phi_3(r)$
- The payoff is obtained after the maximum duration of stages 1 & 3!

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

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

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

Conclusion

- We obtain exact, closed-form expressions for the moments of the NPV of serial projects

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

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
Benz