The preemptive stochastic resource-constrained project scheduling problem: An efficient optimal solution procedure

Stefan Creemers

(December 5, 2016)

IÉSEG SCHOOL OF MANAGEMENT

KATHOLIEKE UNIVERSITEIT LEUVEN

Agenda

- Past work
- New approach
- What about the SRCPSP?
- Contribution

Agenda

- Past work
- New approach
- What about the SRCPSP?
- Contribution

Past work: overview

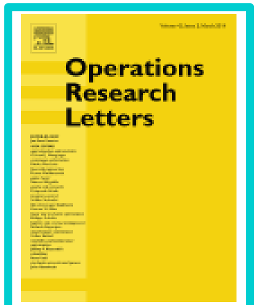
Creemers, Leus, Lambrecht (2010). Scheduling Markovian PERT networks to maximize the net present value, Operations Research Letters.

Past work: overview

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

1. Maximum-eNPV objective
2. No resources
3. Exponentially-distributed activity durations
4. Use of a SDP recursion to obtain the optimal policy

Past work: overview

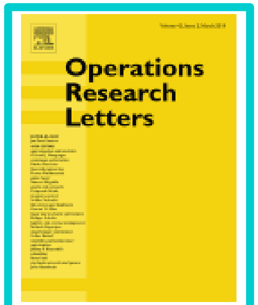


Creemers, Leus, Lambrecht (2010). Scheduling Markovian PERT networks to maximize the net present value, Operations Research Letters.

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

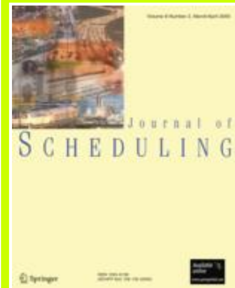
1. Maximum-eNPV objective
2. No resources
3. Exponentially-distributed activity durations
4. Use of a SDP recursion to obtain the optimal policy

Past work: overview



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

1. Maximum-eNPV objective
2. No resources
3. Exponentially-distributed activity durations
4. Use of a SDP recursion to obtain the optimal policy



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

1. Minimum-makespan objective
2. Renewable resources
3. General activity durations (PH approximation)
4. Use of an improved/modified SDP recursion

Agenda

- Past work
- New approach
- What about the SRCPSP?
- Contribution

New approach

PAST WORK

1. SDP recursion
2. Optimal solution
3. General activity durations
4. eNPV & SRCPSP
5. UDCs to structure state space
6. Upper bound state space = 3^n

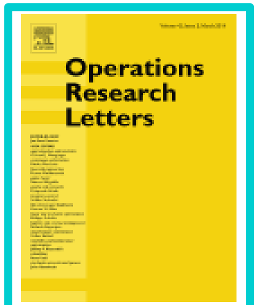
New approach

PAST WORK

1. SDP recursion
2. Optimal solution
3. General activity durations
4. eNPV & SRCPSP
5. UDCs to structure state space
6. Upper bound state space = 3^n

Main bottleneck = memory!

New approach



PAST WORK

1. SDP recursion
2. Optimal solution
3. General activity durations
4. eNPV & SRCPSP
5. UDCs to structure state space
6. Upper bound state space = 3^n

Main bottleneck = memory!

?

NEW APPROACH

1. SDP recursion
2. Optimal solution
3. General activity durations
4. eNPV & SRCPSP
5. No UDCs
6. Upper bound state space = 2^n

New approach: results

- Computational experiment to compare the old and the new approach with respect to:
  - The number of instances solved
  - The computation speed (CPU times)
  - The average maximum number of states stored in memory
- We use a dataset with 30 projects for each:
  - Number of activities (*n* between 10 & 70)
  - Order Strength (OS equal to 0.8, 0.6, and 0.4)

New approach: number of instances solved

OLD			
Number 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	16
n = 60	30	30	0
n = 70	30	29	0

New approach: number of instances solved

OLD			
Number 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	16
n = 60	30	30	0
n = 70	30	29	0

NEW			
Number 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 approach: average CPU time (sec)

OLD			
Average CPU time (sec)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	0.00	0.00
n = 30	0.00	0.00	0.00
n = 40	0.00	0.00	41.1
n = 50	0.00	3.02	899
n = 60	0.00	39.4	NA
n = 70	0.00	365	NA

New approach: average CPU time (sec)

OLD			
Average CPU time (sec)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	0.00	0.00
n = 30	0.00	0.00	0.00
n = 40	0.00	0.00	41.1
n = 50	0.00	3.02	899
n = 60	0.00	39.4	NA
n = 70	0.00	365	NA

NEW			
Average CPU time (sec)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	0.00	0.00
n = 30	0.00	0.00	0.00
n = 40	0.00	0.00	12.3
n = 50	0.00	0.00	270
n = 60	0.00	6.57	8960
n = 70	0.00	61.2	195691

On average, we improve computation times by a factor of 180!

New approach:
average maximum number of states

OLD			
Average maximum # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	2.39	38.6
n = 30	0.00	24.8	934
n = 40	2.9	273	25413
n = 50	9.97	2155	315807
n = 60	37.9	21140	NA
n = 70	112	149925	NA

New approach: average maximum number of states

OLD			
Average maximum # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	2.39	38.6
n = 30	0.00	24.8	934
n = 40	2.9	273	25413
n = 50	9.97	2155	315807
n = 60	37.9	21140	NA
n = 70	112	149925	NA

NEW			
Average maximum # states (x1000)			
	OS = 0.8	OS = 0.6	OS = 0.4
n = 10	0.00	0.00	0.00
n = 20	0.00	0.00	0.00
n = 30	0.00	0.00	2.87
n = 40	0.00	1.28	30.4
n = 50	0.00	4.87	210
n = 60	0.00	20.2	1693
n = 70	0.00	79.1	11006

On average, we reduce memory requirements by a factor of 364!

Agenda

- Past work
- New approach
- What about the SRCPSP?
- Contribution

SRCPSP results: computational performance

	J30	
	Old	New
Instances in set	480	480
Instances solved	480	480
Average CPU time (sec)	0.48	0.02
Average max # states (x1000)	176	1.99

SRCPSP results: computational performance

	J30		J60	
	Old	New	Old	New
Instances in set	480	480	480	480
Instances solved	480	480	303	303 (480)
Average CPU time (sec)	0.48	0.02	1591	81.6
Average max # states (x1000)	176	1.99	374499	508

SRCPSP results: computational performance

	J30		J60	
	Old	New	Old	New
Instances in set	480	480	480	480
Instances solved	480	480	303	303 (480)
Average CPU time (sec)	0.48	0.02	1591	81.6
Average max # states (x1000)	176	1.99	374499	508

We are even able to solve 196 instances of the J90 dataset and 3 instances of the J120 dataset

Agenda

- Past work
- New approach
- What about the SRCPSP?
- Contribution

Contributions

We improve the models of Creemers et al. (2010) and Creemers (2015) and obtain an increase in computational efficiency with factor 180 and a reduction of memory requirements with factor 364!

Contributions

We improve the models of Creemers et al. (2010) and Creemers (2015) and obtain an increase in computational efficiency with factor 180 and a reduction of memory requirements with factor 364!

We can use our model to find the optimal expected NPV for projects with up to 120 activities that have general activity durations!

Contributions

We improve the models of Creemers et al. (2010) and Creemers (2015) and obtain an increase in computational efficiency with factor 180 and a reduction of memory requirements with factor 364!

We can use our model to find the optimal expected NPV for projects with up to 120 activities that have general activity durations!

Our model can also be used to study the SRCPSP where the execution of activities is allowed to be interrupted (i.e., we can assess the value of splitting activities).

?