Technical Appendix

Technical appendix of the paper "An analysis of network and resource indicators for resource-constrained project scheduling problem instances"

Reference: Vanhoucke, M. and Coelho, J. (2021). An analysis of network and resource indicators for resource-constrained project scheduling problem instances. Computers and Operations Research, 132, 105260
(doi: 10.1016/j.cor.2021.105260)

Abstract

This technical appendix discusses an illustrative example to shows the detailed steps of our search algorithm to transform an instance into two new equivalent instances. The original is published in Computers and Operations Research and introduces a new *instance equivalence* concept to show that instances might have very different values for their resource indicators without changing any possible solution for this instance. The concept is based on four theorems and a search algorithm that transforms existing instances into new equivalent instances with more compact resources.

This technical appendix contains two sections. First, the project example is shown using an activity-on-the-node network and relevant resource data. Secondly, the detailed steps of the search algorithm to transform the project instance into two new equivalent instances (lowRU and highRD) is shown.

Note: Some references to figure and table numbers are given that do not appear in this Technical Appendix but are only displayed in the original paper version.

Project example

Figure 3 shows the example network instance 259 of the DC1 set (Vanhoucke et al., 2016; Vanhoucke and Coelho, 2018). The network and resource data is given in Table 6 and shows that the project makes use of four renewable resources with availabilities of 6, 9, 11 and 10 respectively. The table also contains the data for the three sets PCA, RCA and CA defined earlier.

Table 6: Network and resource data for example project instance of Figure 3

		Network		Resources				Sets		
i	d_i	$Suc(i)$	$Prec(i)$	1	2	3	4	$PCA(i)$	$RCA(i)$	$CA(i)$
1	0	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12							2,3,4,5,6,7,8,9,10,11,12	
2	5	5, 6, 8, 9, 10, 11, 12	1			2		3, 4, 7	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	3, 4, 7
3	5	5, 6, 8, 9, 10, 11, 12	1			7		2, 4, 7	1, 2, 4, 5, 6, 8, 9, 10, 11, 12	2, 4
4	2	5, 6, 7, 8, 9, 10, 11, 12	1	1				2, 3	1, 2, 3, 5, 6, 7, 9, 10, 11, 12	2, 3
5	6	8, 9, 10, 12	1, 2, 3, 4				6	6, 7, 11	1, 2, 3, 4, 7, 8, 9, 11, 12	7, 11
6	4	9, 10, 12	1, 2, 3, 4				8	5, 7, 8, 11	1, 2, 3, 4, 7, 8, 9, 11, 12	7, 8, 11
7	4	8, 9, 10, 11, 12	1, 4			7		2, 3, 5, 6	1, 2, 4, 5, 6, 8, 9, 10, 11, 12	2, 5, 6
8	1	9, 10, 12	1, 2, 3, 4, 5, 7	6				6, 11	1, 2, 3, 5, 6, 7, 9, 10, 11, 12	6, 11
9	4	10, 12	1, 2, 3, 4, 5, 7, 6, 8		7			11	1, 2, 3, 4, 5, 6, 7, 8, 10, 12	
10	7	12	1, 2, 3, 4, 5, 7, 6, 8, 9				8	11	1, 2, 3, 4, 7, 8, 9, 11, 12	11
11	2	12	1, 2, 3, 4, 7		9			5, 6, 8, 9, 10	1, 2, 3, 4, 5, 6, 7, 8, 10, 12	5, 6, 8, 10
12	0		1, 2, 3, 4, 5, 7, 6, 8, 9, 10, 11						1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	
				6	9	11	10			

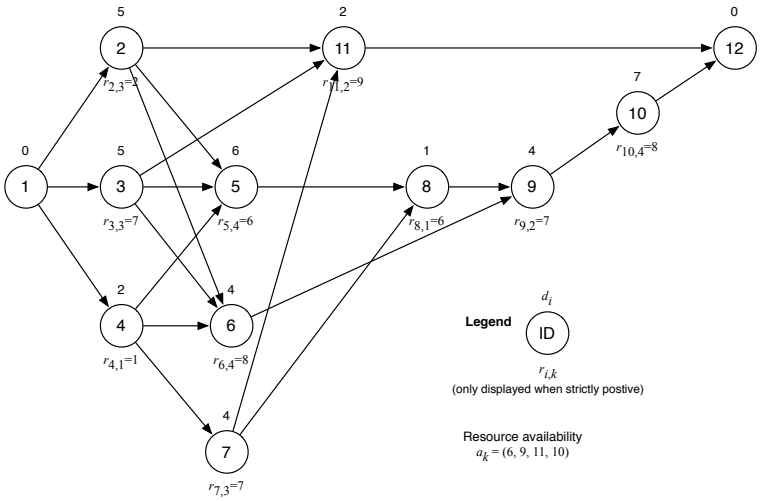

Figure 3: An example project network (instance mv259.rcp of DC1)

Illustrative example of search algorithm

This section displays the calculations of the algorithm of Figure 1 on the example project of Figure 3 and Table 6. The two newly found instances are shown in Table 7 which displays the project network data and resource data of the original instance, as well as the two newly found equivalent instances in the last two columns under heading "New". The calculations of the search of Figure 1 will be illustrated along the remaining paragraphs of this section.

Table 7: Original instance and two new instances of Figure 3

Project network data					Resource data				New (a)	New (b)
i	d_i	$Suc(i)$	$Prec(i)$	$CA(i)$	1	2	3	4	1	1
1	0	2, 3, 4								
2	5	5, 6, 11	1	3, 4, 7			2			
3	5	5, 6, 11	1	2, 4			7		5	5
4	2	5, 6, 7	1	2, 3	1					
5	6	8	2, 3, 4	7, 11				6	4	4
6	4	9	2, 3, 4	7, 8, 11				8	4	4
7	4	8, 11	4	2, 5, 6			7		1	1
8	1	9	5, 7	6, 11	6					
9	4	10	6, 8			7			5	5
10	7	12	9	11				8		4
11	2	12	2, 3, 7	5, 6, 8, 10		9			1	1
12	0		10, 11							
				a_k =	6	9	11	10	5	5

$iter = 0$: Theorems 1, 2 and 3 [Table 8]

The algorithm of Figure 1 starts by sequentially and iteratively applying Theorems 1,

2 and 3 until no further changes can be found. Table 8 displays a summary of all changes in the original instance with four resources to a reduced instance with only 2 resources. First, Theorem 1 (T1-IRD) is applied three times, removing the resource demand for activity 4 (resource 1), activity 8 (resource 1) and activity 10 (resource 4) respectively. Theorem 2 (T2-DR) then removes resource 1 since it is dominated by resource 2. Finally, Theorem 3 (T3-JR) joins two resources into a single resource, resulting in a reduced project instance with only renewable resources with availabilities 99 and 10 respectively. In a final step, displayed in the last two columns of Table 8, the second resource (originally resource 4) is rescaled (SCALE) by dividing the requirements and availabilities by 2. This reduced instance is now input for the next iteration.

Table 8: $iter = 0$: Theorems (1, 2, 3)

	Original				T1-IRD				T1-IRD				T1-IRD				T2-DR			T3-JR		SCALE	
i	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	2	3	4	2/3	4	2/3	4
2			2				2				2				2			2		18		18	
3			7				7				7				7			7		63		63	
4	1				0																		
5				6				6				6				6			6		6		3
6				8				8				8				8			8		8		4
7			7				7				7				7			7		63		63	
8	6				6				0														
9		7				7				7				7			7			77		77	
10				8				8				8				0							
11		9				9				9				9			9			99		99	
	6	9	11	10	6	9	11	10	6	9	11	10	6	9	11	10	9	11	10	99	10	99	5

iter = 1**:** Algorithm 1 (`SaturateResources`) + Theorems 1, 2 and 3 [Table 9]

The algorithm of Figure 1 continues with the next phase which consists of running Algorithm 1 followed by the iterative calls to the 3 theorems.

Algorithm 1 iterates over all activities in the project network after sorting them in decreasing order of $\sum_k r_{i,k}/a_k$ using the last two columns of Table 8 (and using the largest d_i as a tie-break) resulting in the sorted activity list {11, 9, 6, 3, 7, 5, 2, 10, 4, 8, 1, 12}. For each activity in this list, Algorithm 2 is called to calculate the $r_{i,k}^{max}$ variable defined by Theorem 4. The resource waste is then defined as $a_k - r_{i,k}^{max}$, and when $r_{i,k}^{max} < a_k$ (positive value for resource waste), the resource requirements $r_{i,k}$ are increased as shown in Algorithm 1. The results for each activity are given in the columns under the heading "Algorithm 1" in Table 9 in order of the sorted activity list (starting with activity 11, then 9, etc.). The table only shows results for activities for which the found resource waste exceeds zero (i.e. $r_{i,k}^{max} < a_k$) and since for the non-dummy activities 6, 2, 4 and 8, no positive waste was found, they are not displayed in the table.

Table 9 shows that the algorithm starts with activity 11. Algorithm 2 starts with the requirements and availabilities found in the previous iteration as shown in the last columns of Table 8 with $a_1 = 99$ and $a_2 = 5$, and changes the resource requirements $r_{11,1} = 99$ and $r_{11,2} = 0$ to $r_{11,1} = 99$ (no change) and $r_{11,2} = 1$ (increased by 1). The increase of the resource demand for the second resource is the result of the search for the minimum waste of Algorithm 2 for which the detailed steps are shown in Figure 4. More specifically, the figure shows the different steps of the `FoundWaste(11,-)` procedure as a search tree in which the activities that are compatible with activity 11 are recursively

added (ADD) or not (NOT) to c. At the start of this search tree, $r_{i,k}^{max}$ is initialized as $r_{11,1}^{max} = 99$ and $r_{11,2}^{max} = 0$. Each node in the search tree displays the compatible activities added to c and the resource waste variable $waste_k$. Nodes for which the $waste_k$ variable is smaller than zero (e.g. node 3) or for which the activities in c are not compatible with each other (e.g. nodes 5, 7, 12, 15 and 19) are pruned and the algorithm backtracks to the next node in the search tree. In other cases, the algorithm continues until all compatible activities of activity 11 are processed (either added to c or not) and then the resulting resource waste is calculated (by updating the $r_{11,k}^{max}$ variable). The first end node found is node 8 with a resource waste of (0,2) which results in $r_{11,1}^{max} = \max\{99-0, 99\} = 99$ and $r_{11,2}^{max} = \max\{5-2, 0\} = 3$. The second end node is node 10 with resource waste (0,1) which results in an update as $r_{11,1}^{max} = \max\{99-0, 99\} = 99$ and $r_{11,2}^{max} = \max\{5-1, 3\} = 4$. The algorithm continues until all end nodes are evaluated and returns the $r_{i,k}^{max}$ values as $r_{11,1}^{max} = 99$ and $r_{11,2}^{max} = 4$ and updates the resource demand for resource 2 from 0 to 1 as discussed earlier (Table 9), i.e. $r_{11,1} = 99 + \max\{0, 99-99\} = 99$ (no update) and $r_{11,2} = 0 + \max\{0, 5-4\} = 1$ (increase by 1 unit).

When Algorithm 1 has evaluated all activities from the sorted list, the search continues with the iterative calls to the 3 theorems (as shown in the columns "Theorems" of Table 9). The table shows that Theorem 1 removes the resource demand of activity 10, and then rescales the resource data for resource 1 by dividing all data by 9.

Table 9: *iter* = 1: `SaturateResources` (A1) + Theorems 1,2,3

	Algorithm 1												Theorems			
	11		9		3		7		5		2/10		T1-IRD		SCALE	
i	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	18		18		18		18		18		18		18		2	
3	63		63		81	5	81	5	81	5	81	5	81	5	9	5
4																
5		3		3		3		3		4		4		4		4
6		4		4		4		4		4		4		4		4
7	63		63		63		81	1	81	1	81	1	81	1	9	1
8																
9	77		99	5	99	5	99	5	99	5	99	5	99	5	11	5
10												4		0		
11	99	1	99	1	99	1	99	1	99	1	99	1	99	1	11	1
	99	5	99	5	99	5	99	5	99	5	99	5	99	5	11	5

iter = 2**:** Algorithm 1 (`SaturateResources`) + Algorithm 3 (`ReduceResources`) + Theorems 1, 2 and 3 [Table 10]

The algorithm of Figure 1 now continues with the next iteration (*iter* = 2) by first calling Algorithm 1 once again (Column 2 and 3 with heading "A1" in Table 10), followed by the procedure of Algorithm 3 (columns "A3") and the iterative calls to the 3 theorems (columns "Theorems").

First, Algorithm 1 changes the resource demand of activity 10 from 0 to 4 using the `FoundWaste` procedure, and its detailed steps are not explained further. The resulting

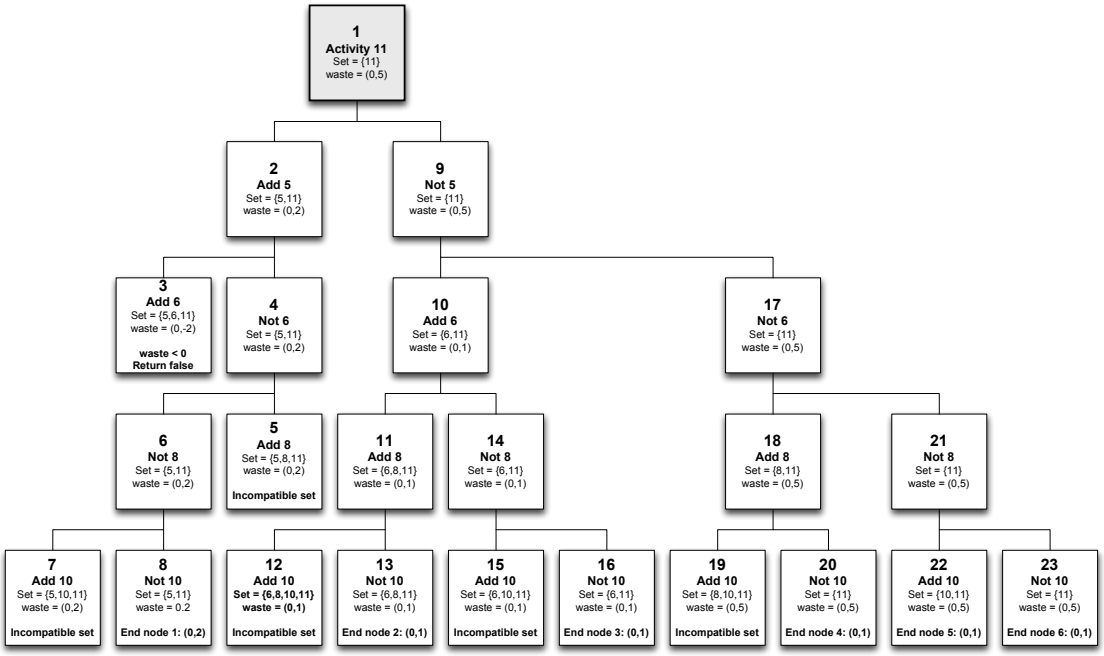

Figure 4: `FoundWaste(11,-)` procedure for activity 11 of Table 9

instance is displayed in columns 2 and 3 of Table 10 and is now the input instance of Algorithm 3. Algorithm 3 is called for the first time during the search and copies this instance I into a temporary instance $I^{current}$ which serves as input for the activity enumeration. It iterates over all activities in the order of increasing activity durations using the sorted list $\{10, 5, 2, 3, 6, 7, 9, 4, 11, 8, 1, 12\}$. In each iteration, the algorithm iteratively changes the instance $I^{current}$ into I^{aux} by putting the resource demand for an activity to zero, and then calls Algorithm 1 to check whether it modifies I^{aux} back into the original instance I. If this is true, the resource demand can be permanently set to zero, and the algorithm continues with the next activity from the list. Table 10 displays the sequential changes for the different activities in the list, and shows only the changes where $r_{i,k}$ values are set to zero (for activities 10, 2, 3 and 9).

The resulting instance is then again subject to the iterative calls of the 3 theorems. Theorem 1 is used twice and puts $r_{7,1} = r_{11,1} = 0$ followed by Theorem 2 that removes the first resource. The resulting instance under column "T2-DR" is the LowRU instance of Figure 1. This instance is also shown as the first new instance (a) in Table 7. The resource indicators for this new instance can be compared with the indicators for the original instance in Table 11.

iter = 3**:** Algorithm 1 (`SaturateResources`) [Last column of Table 10]

In a final iteration, the algorithm of Figure 1 calls Algorithm 1 for the very last time to obtain the "highRD" instance which is displayed in the last column of Table 10.

Table 10: *iter* = 2+3: `SaturateResources` (A1) + `ReduceResources` (A3) + Theorems 1,2,3

	A1		A3								Theorems					A1
			10		2		3		9		T1-IRD		T1-IRD		T2-DR	
i	1	2	1	2	1	2	1	2	1	2	1	2	1	2	2	2
2	2		2		0		0		0							
3	9	5	9	5	9	5	0	5	0	5		5		5	5	5
4																
5		4		4		4		4		4		4		4	4	4
6		4		4		4		4		4		4		4	4	4
7	9	1	9	1	9	1	9	1	9	1	0	1		1	1	1
8																
9	11	5	11	5	11	5	11	5	0	5		5		5	5	5
10		4		0		0		0		0						4
11	11	1	11	1	11	1	11	1	11	1	11	1	0	1	1	1
	11	5	11	5	11	5	11	5	11	5	11	5	11	5	5	5

This instance is also shown as the second new instance (b) in Table 7 and its resource indicators are displayed in Table 11. Note that the two new instances, lowRU and highRD, can be used to calculate the $CA(i)$ values, which would be exactly the same as the values found in Table 6, showing that the instances impose the same resource restrictions as the original instance.

Table 11: Resource indicators for three equivalent instances

Instance	#	RF	RU	RS	RC
Original	4	0.250	1	0.694	0.659
lowRU (a)	1	0.6	0.6	0	0.666
highRD (b)	1	0.7	0.7	0	0.685