10.1287/msom.1070.0132

ON-LINE APPENDIX A
Design of Numerical Experiments

Our numerical study includes different scenarios generated by the combination of the parameters presented in Table A1.

Table A1. Scenarios for Number of Machines, Machine Availabilities, and Downtime Costs.

Scenarios for Costs		**Scenarios for Number of Machines**		**Scenarios for Machine Availabilities**	
#	(c_1, c_2, c_3, c_4)	#	(N_1, N_2, N_3, N_4)	#	$(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$
1	(1.0,1.0,1.0,1.0)	1	(5, 5, 5, 5)	1	(95%,90%,85%,75%)
2	(0.1,0.3,0.6,1)	2	(2,4,6,8)	2	(75%,85%,90%,95%)
3	(1,0.6,0.3,0.1)	3	(8,6,4,2)	3	(80%,80%,80%,80%)
4	(0.1,0.6,0.8,0.9)				
5	(0.9,0.8,0.6,0.1)				

Number of Machines: We consider systems with 4 different types of machines in which the total number of machines in the system is 20. The motivation behind choosing systems with four machine types is that neither systems with two machine types nor systems with three machine types accommodate the sufficient complexity that is required to evaluate the effect of different machine priority and repairmen assignment rules. On the other hand, analysis of systems with more than four machine types requires solving problems with large state spaces (that are computationally intensive) while not providing much additional insight compared with systems with four machine types. The scenarios for the number of machines include cases where all machine types have an equal or unequal number of machines. Scenarios with an unequal number of machines are chosen such that some machine types have four times, three times, or twice as many machines as other machine types. Note that each scenario for the number of machines in fact generates a maximum of $4! = 24$ cases in our numerical study, depending on how machine are prioritized. For example, if the optimal static machine priority policy dictates that the first, second, third, and fourth priorities should be given to machine types 3, 4, 1, and 2, respectively, then scenario 2 for the number of machines of each type will result in a system in which $(N_1, N_2, N_3, N_4) = (6, 8, 2, 4)$.

Machine Downtime Costs: Our set of experiments include five different scenarios for machine downtime costs. We chose down time costs to be between 0.1 and 1. (Note that the relative values (not the absolute values) of the downtime costs are important. Scenarios for machine downtime costs are chosen such that machines have the same downtime costs, or when costs are different, the difference between c_i and c_{i+1} increases (or decreases) in a linear or nonlinear fashion (i.e., $|c_i - c_{i+1}|$ increases or decreases in a linear or nonlinear fashion).

Machine Availability and Failure and Repair Rate: Based on the results of several numerical study that we did, we found that considering machine availabilities is enough to incorporate the effect of machine failure and repair rates. This is because availabilities carry information about both machine failure and repair rates (i.e., $\alpha_i = \mu_i/[\lambda_i + \mu_i]$). Therefore, in our set of experiments we chose to concentrate on the machine availabilities. The three scenarios of machine availabilities in Table 1 refer to systems with equal and unequal machine availabilities. The case with unequal availability, the availabilities are increasing or decreasing with the machine index and are designed to include availabilities ranging from 0.75 to 0.95. Note that machines with availabilities less than 0.75 are not common in practice.

Repairmen skill set matrices: One important issue in choosing skill set matrices is that they should be different such that our repairman assignment rules result in different assignments under the same skill set matrix. Therefore, we chose skill set matrices that result in different repairman assignments under our repairman-assignment rules. Furthermore, we added well-known skill set matrices such as chain, and pyramid to our set of matrices. The pyramid skill set matrix is representative of a repair crew that includes repairmen with different amounts of experience, including repairmen with one, two, three, and four skills. In order to also consider the effects of the total number of skills, we choose cases with a total number of 8, 9, 10, and 12, skills. The skill set matrices are as follows:

$$M_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad M_3 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \qquad M_5 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

ON-LINE APPENDIX B
The Detail Data for Chain Skill Set Study

B.1. Systematic Approach:

Our systematic approach generated 60 cases for systems with three machines and three repairmen. These 60 cases were generated as follows. We set $c_i N_i(1-\alpha_i) = Z$ for all $i = 1, 2, 3$. We considered five different values 0.1, 0.5, 1.0, 1.5 and 2 for Z, and three different values $(5, 5, 5)$ or $(2, 4, 9)$ or $(9, 4, 2)$ for (N_1, N_2, N_3). Then, for each choice of Z and (N_1, N_2, N_3) we either: (i) set availabilities to be $\alpha_1 = 0.65, \alpha_2 = 0.75, \alpha_3 = 0.95$ and obtained downtime cost (c_1, c_2, c_3) that results in our choice of Z, or (ii) set (c_1, c_2, c_3) to $(1.4, 1.0.6)$ or $(0.6, 1, 1.4)$ and obtained availabilities $(\alpha_1, \alpha_2, \alpha_3)$ that results in our choice of Z. This process resulted in 60 cases which are presented in Table B1.

Table B1. Cases Generated through *Systematic* Approach

(c_1, c_2, c_3)	(N_1, N_2, N_3)	$(\lambda_1, \lambda_2, \lambda_3)$	(μ_1, μ_2, μ_3)	$cN(1-\alpha)$
(0.06, 0.08, 0.4)	(5, 5, 5)	(2, 4, 8)	(3.71, 12, 152)	0.1
(1.4, 1, 0.6)	(5, 5, 5)	(2, 4, 8)	(138, 196, 232)	0.1
(0.14, 0.1, 0.22)	(2, 4, 9)	(2, 4, 8)	(3.71, 12, 152)	0.1
(1.4, 1, 0.6)	(2, 4, 9)	(2, 4, 8)	(54, 156, 424)	0.1
(0.03, 0.1, 1)	(9, 4, 2)	(2, 4, 8)	(3.71, 12, 152)	0.1
(1.4, 1, 0.6)	(9, 4, 2)	(2, 4, 8)	(250, 156, 88)	0.1
(0.06, 0.08, 0.4)	(5, 5, 5)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.1
(1.4, 1, 0.6)	(5, 5, 5)	(2.9, 2.86, 2.76)	(200, 140, 80)	0.1
(0.14, 0.1, 0.22)	(2, 4, 9)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.1
(1.4, 1, 0.6)	(2, 4, 9)	(7.41, 3.59, 1.51)	(200, 140, 80)	0.1
(0.03, 0.1, 1)	(9, 4, 2)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.1
(1.4, 1, 0.6)	(9, 4, 2)	(1.6, 3.59, 7.27)	(200, 140, 80)	0.1
(0.29, 0.4, 2)	(5, 5, 5)	(2, 4, 8)	(3.71, 12, 152)	0.5
(1.4, 1, 0.6)	(5, 5, 5)	(2, 4, 8)	(26, 36, 40)	0.5
(0.71, 0.5, 1.11)	(2, 4, 9)	(2, 4, 8)	(3.71, 12, 152)	0.5
(1.4, 1, 0.6)	(2, 4, 9)	(2, 4, 8)	(9.2, 28, 78.4)	0.5
(0.16, 0.5, 5)	(9, 4, 2)	(2, 4, 8)	(3.71, 12, 152)	0.5
(1.4, 1, 0.6)	(9, 4, 2)	(2, 4, 8)	(48.4, 28, 11.2)	0.5
(0.29, 0.4, 2)	(5, 5, 5)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.5
(1.4, 1, 0.6)	(5, 5, 5)	(15.38, 15.56, 16)	(200, 140, 80)	0.5
(0.71, 0.5, 1.11)	(2, 4, 9)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.5
(1.4, 1, 0.6)	(2, 4, 9)	(43.48, 20, 8.16)	(200, 140, 80)	0.5
(0.16, 0.5, 5)	(9, 4, 2)	(107.69, 46.67, 4.21)	(200, 140, 80)	0.5
(1.4, 1, 0.6)	(9, 4, 2)	(8.26, 20, 57.14)	(200, 140, 80)	0.5
(0.57, 0.8, 4)	(5, 5, 5)	(2, 4, 8)	(3.71, 12, 152)	1.0
(1.4, 1, 0.6)	(5, 5, 5)	(2, 4, 8)	(12, 16, 16)	1.0
(1.43, 1, 2.22)	(2, 4, 9)	(2, 4, 8)	(3.71, 12, 152)	1.0
(1.4, 1, 0.6)	(2, 4, 9)	(2, 4, 8)	(3.6, 12, 35.2)	1.0
(0.32, 1, 10)	(9, 4, 2)	(2, 4, 8)	(3.71, 12, 152)	1.0
(1.4, 1, 0.6)	(9, 4, 2)	(2, 4, 8)	(23.2, 12, 1.6)	1.0
(0.57, 0.8, 4)	(5, 5, 5)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.0
(1.4, 1, 0.6)	(5, 5, 5)	(33.33, 35, 40)	(200, 140, 80)	1.0
(1.43, 1, 2.22)	(2, 4, 9)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.0
(1.4, 1, 0.6)	(2, 4, 9)	(111.11, 46.67, 18.18)	(200, 140, 80)	1.0
(0.32, 1, 10)	(9, 4, 2)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.0
(1.4, 1, 0.6)	(9, 4, 2)	(17.24, 46.67, 400)	(200, 140, 80)	1.0

Table B1 (Continued). Cases Generated through *Systematic* Approach

(c_1, c_2, c_3)	(N_1, N_2, N_3)	$(\lambda_1, \lambda_2, \lambda_3)$	(μ_1, μ_2, μ_3)	$cN(1-\alpha)$
(0.86, 1.2, 6)	(5, 5, 5)	(2, 4, 8)	(3.71, 12, 152)	1.5
(1.4, 1, 0.6)	(5, 5, 5)	(2, 4, 8)	(7.33, 9.33, 8)	1.5
(2.14, 1.5, 3.33)	(2, 4, 9)	(2, 4, 8)	(3.71, 12, 152)	1.5
(1.4, 1, 0.6)	(2, 4, 9)	(2, 4, 8)	(1.73, 6.67, 20.8)	1.5
(0.48, 1.5, 15)	(9, 4, 2)	(2, 4, 8)	(3.71, 12, 152)	1.5
(0.6, 1, 1.4)	(9, 4, 2)	(2, 4, 8)	(5.2, 6.67, 6.93)	1.5
(0.86, 1.2, 6)	(5, 5, 5)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.5
(1.4, 1, 0.6)	(5, 5, 5)	(54.55, 60, 80)	(200, 140, 80)	1.5
(2.14, 1.5, 3.33)	(2, 4, 9)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.5
(1.4, 1, 0.6)	(2, 4, 9)	(230.77, 84, 30.77)	(200, 140, 80)	1.5
(0.48, 1.5, 15)	(9, 4, 2)	(107.69, 46.67, 4.21)	(200, 140, 80)	1.5
(0.6, 1, 1.4)	(9, 4, 2)	(76.92, 84, 92.31)	(200, 140, 80)	1.5
(1.14, 1.6, 8)	(5, 5, 5)	(2, 4, 8)	(3.71, 12, 152)	2.0
(1.4, 1, 0.6)	(5, 5, 5)	(2, 4, 8)	(5, 6, 4)	2.0
(2.86, 2, 4.44)	(2, 4, 9)	(2, 4, 8)	(3.71, 12, 152)	2.0
(1.4, 1, 0.6)	(2, 4, 9)	(2, 4, 8)	(0.8, 4, 13.6)	2.0
(0.63, 2, 20)	(9, 4, 2)	(2, 4, 8)	(3.71, 12, 152)	2.0
(0.6, 1, 1.4)	(9, 4, 2)	(2, 4, 8)	(3.4, 4, 3.2)	2.0
(1.14, 1.6, 8)	(5, 5, 5)	(107.69, 46.67, 4.21)	(200, 140, 80)	2.0
(1.4, 1, 0.6)	(5, 5, 5)	(80, 93.33, 160)	(200, 140, 80)	2.0
(2.86, 2, 4.44)	(2, 4, 9)	(107.69, 46.67, 4.21)	(200, 140, 80)	2.0
(1.4, 1, 0.6)	(2, 4, 9)	(500, 140, 47.06)	(200, 140, 80)	2.0
(0.63, 2, 20)	(9, 4, 2)	(107.69, 46.67, 4.21)	(200, 140, 80)	2.0
(0.6, 1, 1.4)	(9, 4, 2)	(117.65, 140, 200)	(200, 140, 80)	2.0

B.2. Random Approach:

Our random approach generated 48 cases as follows. We set $c_i N_i(1-\alpha_i) = Z$ for all $i = 1, 2, 3$, and we considered 48 different values of Z. These values are almost uniformly distributed between 0.1 and 2. For each value of Z, through random generation and trial and error, we generated c_i, N_i, λ_i and μ_i for $i = 1, 2, 3$ that: (i) result in machine availabilities greater than 50%, (ii) total number of machines less than 24, (iii) number of machines of each type less than 14, and (iv) downtime costs between 0.1 and 4. Table B2 depicts the 48 cases that we generated through our random approach. Note that half of the scenarios in Table B2 corresponds to cases where $c_i = 1$ for all $i = 1, 2, \ldots, N$. These cases represents situations where the objective is to minimize the total average number of broken machines.

Table B2. Cases Generated through *Random* Approach

(c_1, c_2, c_3)	(N_1, N_2, N_3)	$(\lambda_1, \lambda_2, \lambda_3)$	(μ_1, μ_2, μ_3)	$cN(1-\alpha)$
(0.1 , 0.15 , 0.6)	(7 , 6 , 5)	(3.01 , 4.24 , 4.41)	(18.07 , 33.93 , 127.88)	0.10
(1.4 , 0.1 , 0.2)	(5 , 5 , 4)	(4.71 , 3.07 , 3.92)	(295.15 , 10.88 , 24.59)	0.11
(0.5 , 0.1 , 0.15)	(9 , 5 , 5)	(1.24 , 4.49 , 1.90)	(35.90 , 10.48 , 7.59)	0.15
(0.1 , 0.1 , 0.8)	(8 , 5 , 4)	(6.70 , 2.06 , 1.37)	(24.83 , 3.99 , 24.39)	0.17
(0.1 , 0.15 , 0.5)	(6 , 5 , 5)	(3.01 , 1.19 , 4.39)	(6.03 , 3.28 , 50.43)	0.20
(0.5 , 0.3 , 0.2)	(8 , 6 , 4)	(3.70 , 5.66 , 1.98)	(60.64 , 38.67 , 4.91)	0.23
(1 , 0.2 , 0.7)	(7 , 8 , 3)	(2.25 , 2.60 , 2.28)	(60.68 , 14.06 , 16.90)	0.25
(1.6 , 0.2 , 0.6)	(10 , 4 , 3)	(9.78 , 1.20 , 2.20)	(591.95 , 2.49 , 13.04)	0.26
(0.2 , 0.6 , 0.9)	(7 , 7 , 3)	(3.41 , 2.76 , 1.06)	(13.05 , 37.23 , 8.80)	0.29
(0.2 , 0.3 , 0.2)	(11 , 4 , 3)	(3.98 , 0.10 , 2.52)	(25.21 , 0.30 , 2.52)	0.30
(1 , 0.5 , 0.6)	(11 , 5 , 3)	(5.27 , 4.23 , 2.91)	(175.87 , 28.84 , 13.44)	0.32
(1 , 0.7 , 0.2)	(7 , 5 , 5)	(6.35 , 0.24 , 1.08)	(120.71 , 2.14 , 2.01)	0.35
(0.5 , 0.2 , 2)	(7 , 7 , 5)	(5.65 , 6.24 , 2.75)	(46.35 , 16.76 , 69.66)	0.38
(1.7 , 0.3 , 0.8)	(8 , 4 , 2)	(4.87 , 0.45 , 0.56)	(156.76 , 0.87 , 1.63)	0.41
(0.6 , 0.2 , 1.2)	(6 , 8 , 2)	(2.97 , 3.89 , 0.22)	(21.34 , 10.25 , 1.00)	0.44
(1 , 1 , 1)	(14 , 3 , 2)	(5.94 , 1.71 , 1.58)	(171.10 , 9.22 , 5.13)	0.47
(0.1 , 0.7 , 4)	(14 , 3 , 2)	(12.53 , 0.83 , 1.28)	(22.56 , 2.64 , 19.19)	0.50
(1 , 1 , 1)	(10 , 4 , 1)	(3.13 , 2.43 , 1.15)	(59.47 , 17.00 , 1.15)	0.50
(1 , 1 , 1)	(8 , 2 , 4)	(5.28 , 1.50 , 1.50)	(74.48 , 4.16 , 9.82)	0.53
(1 , 1 , 1)	(6 , 2 , 5)	(4.77 , 2.31 , 2.53)	(46.34 , 5.95 , 20.08)	0.56
(1 , 1 , 1)	(7 , 2 , 5)	(4.44 , 3.53 , 1.07)	(48.27 , 8.43 , 7.98)	0.59
(1 , 1 , 1)	(9 , 6 , 2)	(1.19 , 2.94 , 3.59)	(16.15 , 25.55 , 7.99)	0.62
(1 , 1 , 1)	(10 , 2 , 5)	(2.39 , 4.80 , 0.95)	(34.33 , 9.96 , 6.33)	0.65
(1 , 1 , 1)	(9 , 5 , 3)	(7.23 , 2.02 , 0.50)	(88.41 , 12.83 , 1.71)	0.68
(1 , 1 , 1)	(8 , 5 , 3)	(2.33 , 1.03 , 1.87)	(23.87 , 6.24 , 6.04)	0.71
(1 , 1 , 1)	(10 , 5 , 2)	(7.10 , 2.00 , 3.45)	(88.90 , 11.51 , 5.87)	0.74
(1.2 , 1 , 0.7)	(6 , 5 , 4)	(3.94 , 2.04 , 2.07)	(33.91 , 11.59 , 5.65)	0.75
(1 , 1 , 1)	(6 , 12 , 4)	(3.41 , 5.47 , 2.24)	(23.18 , 79.84 , 9.40)	0.77
(1 , 1 , 1)	(12 , 4 , 3)	(2.00 , 2.00 , 1.50)	(28.00 , 8.00 , 4.13)	0.80
(1 , 1 , 1)	(7 , 12 , 3)	(3.33 , 6.92 , 2.92)	(24.79 , 93.14 , 7.63)	0.83
(1 , 1 , 1)	(10 , 3 , 4)	(6.35 , 2.03 , 3.04)	(67.52 , 5.04 , 11.09)	0.86
(1 , 1 , 1)	(11 , 5 , 3)	(6.05 , 3.00 , 2.96)	(68.74 , 13.85 , 7.01)	0.89
(1 , 1 , 1)	(7 , 5 , 12)	(5.15 , 1.65 , 0.70)	(34.05 , 7.31 , 8.43)	0.92
(1 , 1 , 1)	(3 , 12 , 5)	(6.89 , 3.91 , 0.62)	(14.87 , 45.44 , 2.65)	0.95
(1 , 1 , 1)	(4 , 2 , 5)	(5.90 , 4.57 , 0.90)	(18.18 , 4.76 , 3.70)	0.98
(2 , 1 , 1.5)	(6 , 8 , 2)	(1.00 , 2.00 , 3.00)	(11.00 , 14.00 , 6.00)	1.00
(1 , 1 , 1)	(8 , 4 , 2)	(7.03 , 2.15 , 1.00)	(48.68 , 6.37 , 0.98)	1.01
(1 , 1 , 1)	(7 , 4 , 3)	(4.99 , 6.67 , 1.33)	(28.57 , 18.99 , 2.51)	1.04
(1 , 1 , 1)	(10 , 5 , 3)	(1.23 , 1.00 , 0.80)	(10.26 , 3.67 , 1.44)	1.07
(0.9 , 1 , 1.1)	(10 , 3 , 4)	(8.39 , 2.00 , 2.42)	(60.26 , 3.45 , 7.27)	1.10
(1 , 1 , 1)	(8 , 5 , 3)	(2.00 , 2.24 , 1.03)	(12.57 , 7.95 , 1.79)	1.10
(1 , 1 , 1)	(3 , 9 , 5)	(6.41 , 1.05 , 4.68)	(10.61 , 7.34 , 16.04)	1.13
(1 , 1 , 1)	(9 , 6 , 4)	(8.80 , 1.40 , 0.85)	(59.46 , 5.86 , 2.07)	1.16
(1 , 1 , 1)	(6 , 12 , 3)	(2.05 , 1.19 , 0.63)	(8.29 , 10.77 , 0.96)	1.19
(0.7 , 2 , 1)	(5 , 5 , 5)	(1.42 , 0.70 , 4.34)	(2.40 , 4.68 , 12.35)	1.30
(0.6 , 1 , 3)	(8 , 7 , 4)	(5.87 , 2.35 , 3.58)	(12.92 , 8.63 , 25.03)	1.50
(0.8 , 0.9 , 2)	(6 , 8 , 4)	(1.45 , 7.95 , 1.74)	(2.64 , 25.72 , 6.47)	1.70
(1 , 4 , 3)	(7 , 6 , 4)	(6.95 , 5.16 , 1.44)	(17.39 , 56.75 , 7.20)	2.00

ON-LINE APPENDIX C
A Myopic Heuristic for Optimal Training Decisions

Although having fully cross-trained repairmen is ideal, training all repairmen for all skills is nevertheless often costly, if not infeasible. Furthermore, Pinker and Shumsky (2000) and Hopp et al. (2002) have shown that adding additional skills often yields diminishing returns in terms of improving the performance of production systems. On the other hand, when a repairman gains more experience by acquiring additional skills, his (her) salary often increases. Consequently, the optimal training decision will reflect the best trade-off between machine downtime cost and repairman cost.

In this section we develop a cost model for the machine-repairman problem that determines the training program which minimizes the cost of both machine downtime and repairmen in a system that follows machine-priority rule $\mathcal{P}$ and repairman-assignment rule $\mathcal{R}$. Let the current skill set matrix be M_0, and let the optimal skill set matrix obtained after implementing the optimal training program be M^*. Also, assume that:

$S_{\mathcal{M}_k}$ = per unit time cost of a repairman who possesses skill set $\mathcal{M}_k$, where $\mathcal{M}_k \subset \{1, 2, \ldots, N\}$.

Therefore, the optimal skill set matrix M^* is the matrix that minimizes $TC(M)$, the total average cost per unit time, where

$$\text{Min} \qquad TC(M) = \sum_{i=1}^{N} c_i E[B_i(M)] + \sum_{k=1}^{K} S_{\mathcal{M}_k} \tag{12}$$

subject to:

$$m_{k,i} = 1 \quad ; \quad \forall m_{k,i} \in \mathcal{E}_{M_0} \tag{13}$$

$$m_{k,i} \leq 1 \quad ; \quad \forall m_{k,i} \notin \mathcal{E}_{M_0}$$

$$\|M\| \leq J \tag{14}$$

$$m_{ki} \in \{0,1\} \quad ; \quad \forall\, k \in \{1, 2, \ldots, K\}; \quad i \in \{1, 2, \ldots, N\}$$

where $\mathcal{E}_{M_0}$ is the set of elements $m_{k,i}$ of current skill set matrix M_0 that are equal to 1. Constraints (13) retain the skills of repairmen from the current skill set matrix M_0 in the optimal skill set matrix M^*. Constraint (14) limits the total number of skills in the optimal skill set matrix M^* to the maximum of J skills ($J \leq KN$). This constraint represents cases where, due to some limitation (e.g., a limited budget for training), a maximum of only $J - \|M_0\|$ skills can be added to the skill set matrix.

Optimization problem (12) suffers from the curse of dimensionality in the following way: Since the relationship between skill set matrix M and the average number of broken machines is very complex, the optimal skill set matrix M^* must be obtained by comparing the cost of various skill set matrices and choosing the matrix that results in the least cost. When the number of repairmen (K) and/or the number of skills (N) are relatively large, the number of skill set matrices that must be examined becomes very large. For example, let us assume that we have five repairmen and five machine types (i.e., $K = N = 5$), and each repairman is skilled in exactly one unique skill (no cross-training, i.e., $\|M_0\| = 5$). Even if we limit the number of skills resulting from additional training to a maximum of five (i.e., $J = 10$), there still exists at least 24,170 choices[3] of skill set matrices that must be evaluated in the objective function (12). This number dramatically increases with the number of repairmen (K) and machine types (N).

We now introduce an algorithm that overcomes the above difficulty and identifies the optimal or near-optimal skill set matrix M^*, by searching among at most $(J - \|M_0\|)NK$ skill set matrices (i.e., 125 skill set matrices for this example).

C.1. Myopic Approach: Instead of searching among all the possible skill set matrices, the myopic approach starts with an initial skill set matrix $M^{(0)} = M_0$, and systematically adds skills to that matrix until adding an additional skill increases the system's total average cost. The myopic approach consists of the following steps:

[3]The 24,170 matrices include 20 matrices with a total of 6 skills each, at least 150 with a total of 7 skills each, at least 1,500 with a total of 8 skills each, at least 7,500 with a total of 9 skills each, and at least 15,000 with a total of 10 skills each.

Algorithm for the Myopic Approach

Step 1: Compute the target cost OC_i for all machine types $i = 1, 2, \ldots, N$, as follows:

$$OC_i = c_i N_i \ (1 - \alpha_i) \qquad \text{where} \qquad \alpha_i = \frac{\mu_i}{\lambda_i + \mu_i}. \tag{15}$$

Step 2: Set the initial skill set matrix $M^{(0)}$ to the current skill set matrix M_0 and set the training counter $j = 0$. Find the next skill that must be added to skill set matrix $M^{(j)}$ as follows:

- **(a)** Solve the machine-repairman problem with the skill set matrix $M^{(j)}$ using the model from Section 3.4 to determine the average number of broken-machines in each type $E[B_i(M^{(j)})]$ for $i = 1, 2, \ldots, N$. Calculate the average broken machine cost per unit time for each machine type $i = 1, 2, \ldots, N$ as $c_i E[B_i(M^{(j)})]$ and the average total cost of the system $TC(M^{(j)})$ using (12).
- **(b)** Choose the machine i^*, where $i^* = argmax_i\{\delta_i\}$ and $\delta_i = c_i E[B_i(M^{(j)})] - OC_i$. The repair skill i^* is the next skill that must be added to skill set matrix $M^{(j)}$.

Step 3: Find the repairman who must be trained for skill i^* as follows:

- **(a)** List all the repairmen who lack the skill to repair machines of type i^*. For each of those repairmen, revise the skill set matrix by assuming that the repairman is trained for skill i^*. Solve the system with the revised skill set matrix, using the model described in Section 3.3 to determine the average number of broken machines, and obtain the total average cost of the system using (12).
- **(b)** Choose the repairman in the list whose training for skill i^* will result in the minimum total average cost. Name the repairman w^*, and let the minimum total average cost be $TC_{w^*}(i^*)$. Go to Step 4.

Step 4: If $TC_{w^*}(i^*) - TC(M^{(j)}) < 0$, then train Repairman w^* in skill i^*. Set the training counter $j = j + 1$, and revise the skill set matrix $M^{(j)}$ to incorporate the training of Repairman w^*. Go to Step 5. However, if $TC_{w^*}(i^*) - TC(M^{(j)}) \geq 0$, then set $\delta_{i^*} = 0$ and return to Step 2b.

Step 5: If $\|M^{(j)}\| = J$, where J is the total number of desired skills, then stop. Otherwise, return to Step 2a.

Step 1 calculates the target cost OC_i which is the minimum downtime cost per unit time possible for machines of type i. This minimum cost is achieved under an unlimited repair capacity such that broken machines of type i never wait for repairmen. Although this may never happen when the number of repairmen is limited, we have nevertheless found this to be a good benchmark for evaluating the opportunity to improve in the downtime cost of each machine.

In Step 2, we evaluate the system with the current skill set, using the model from Section 3.3 to determine the average number of broken machines for each machine type. Then we compare $c_i E[B_i(M^{(j)})]$, the downtime cost of each machine type, with skill set matrix $M^{(j)}$, whose target cost is OC_i. Finally, the repair skill for the machine type that has the largest difference of $\delta_i = c_i E[B_i(M^{(j)})] - OC_i$ is chosen as the next skill to be added to the skill set matrix. The reason is that a larger δ_i often implies that adding a skill will result in more opportunity for downtime cost reduction.

Step 3 evaluates all the possible repairmen who could be trained in the required skill i^*, in order to find the repairman choice that minimizes the total average cost per unit time of the system. Step 4 first compares the average total cost per unit time of the system before and after the training to evaluate the expected savings, and then makes the training decision. If the training chosen does not reduce the total cost per unit time, this process is repeated from Step 2a with the required skill corresponding to the next maximum value of δ_i. Step 5 checks the limit on the maximum number of total skills in the skill set matrix $M^{(j)}$ to satisfy constraint (14) in optimization problem (12).

As we mentioned before, the myopic approach significantly reduces the number of skill set matrices that need to be evaluated to find the optimal skill set matrix M^*. Our numerical study below will show that when the myopic approach does not result in the optimal skill set matrix, the cost difference between employing its suboptimal skill set matrix and the optimal skill set matrix is insignificant.

C.2. Evaluation of the Myopic Approach

In this section we evaluate the performance of our myopic approach. We consider 540 test cases of systems with $N = 4$ machine types and $K = 4$ repairmen, including: 3 scenarios for machine failure and repair rates, 3 scenarios for the number of machines in each type, 4 scenarios for repairman costs, 3 repairman-assignment rules, and 5 scenarios for machine downtime costs. The 3 scenarios for the number of machines are the same as Scenarios #1, #2 and #3 in Table 3. The failure and repair rate scenarios are as shown in Table C1 and the three repairman-assignment rules are LSR, LLP, and LRR.

Table C1. Scenarios for Failure and Repair Rates.

	Type 1		Type 2		Type 3		Type 4	
#	(λ_1, μ_1)	Av_1	(λ_2, μ_2)	Av_2	(λ_3, μ_3)	Av_3	(λ_4, μ_4)	Av_4
1	(1,19)	95%	(4,36)	90%	(15,85)	85%	(10,30)	75%
2	(10,30)	75%	(15,85)	85%	(4,36)	90%	(1,19)	95%
3	(20,80)	80%	(30,120)	80%	(40,160)	80%	(15,60)	80%

Table C2. Scenarios for Machine Downtime and Repairman Base Skill Costs.

Scenario	Machine Downtime Costs				Scenario	Base Skill Costs			
#	c_1	c_2	c_3	c_4	#	S_1	S_2	S_3	S_4
1	100	95	90	85	1	100	90	80	70
2	50	40	30	20	2	50	35	25	10
3	100	90	20	5	3	30	20	12	5
4	50	50	50	50	4	14	6	80	2
5	500	450	400	350					

In Table C2, the machine downtime costs c_i are set to be decreasing in i. After all, one of the main reasons that a machine type is assigned a high priority is because its downtime is more expensive than that of other machine types. The base skill cost S_i in Table C2 is the repairman cost per unit time for those repairmen who are not cross-trained and have only skill i. We also assume that S_i is decreasing in i, since lower-priority machines are often less expensive, so their repair is simpler and requires less experience and training.

Before we present the results of our numerical study, we will first describe how we calculated the repairman cost for a repairman who has more than one skill. For our problem with $N = 4$ types of machines (skills), we calculated each repairman cost according to the following rules:

- If a repairman has two skills $\mathcal{M}_k = \{u, v\}$, where $u < v$, then the repairman cost is $S_{\mathcal{M}_k} = S_u + 0.2S_v$.
- If a repairman has three skills $\mathcal{M}_k = \{u, v, l\}$, where $u < v < l$, then the repairman cost is $S_{\mathcal{M}_k} = S_u + 0.2S_v + 0.1S_l$.
- If a repairman has four skills $\mathcal{M}_k = \{u, v, l, w\}$, where $u < v < l < w$, then the repairman cost is $S_{\mathcal{M}_k} = S_u + 0.2S_v + 0.1S_l + 0.05S_w$.

The repairman cost computation above has two features: (i) it puts more emphasis on the repair skill needed for the higher-priority machines, and (ii) it shows the diminishing returns with respect to the number of skills. Note that we use the above approach to simplify the repairman cost calculations when s/he acquires one more skill. However, the myopic approach can be used under any repairman cost computation method.

In order to evaluate the ability of our myopic approach to obtain the optimal or suboptimal training program, we find the optimal skill set matrix M^* for each of the 540 test cases by comparing the total cost of all possible skill set matrices for each case and choosing the optimal one. We set the current skill set matrix to be a unit matrix (i.e., no cross-training) and we look for the optimal skill set matrix with the maximum number of total skills of $J = 8$. In other words, we are trying to find the best skill set matrix for the system after at most 4 trainings. We examined all 667 possible skill set matrices that had skills ranging from 4 to 8 and found the optimal skill set matrix M^*. Note that the optimal skill set matrix does not necessarily have a total of 8 skills.

For each of the 540 scenarios, we also used our myopic algorithm and obtained the skill set matrix M_{myp}. If our myopic algorithm results in a matrix different from the optimal skill set matrix M^*, then we calculate

the relative error Δ_{myp} as follows:

$$\Delta_{myp} = \frac{TC(M_{myp}) - TC(M^*)}{TC(M^*)}$$

Table C3 summarizes the results for systems under the LSR, LLP, and LRR repairman-assignment rules. Note that in Table C3, the column $\#Opt$ represents the number of cases out of 180 (540/3=180 for each repairman-assignment rule) in which the myopic approach resulted in the optimal training (optimal skill matrix M^*). As Table C3 shows, the myopic approach results in the skill set matrix(M_{myp}) with a total average cost per unit time that is very close to the cost of the optimal skill set matrix(M^*). Hence, we conclude that the myopic approach works well for the systems considered.

Table C3. Evaluation of Myopic Approach.

		Δ_{myp}	
Assignment Rule	$\#Opt$	Avg	Max
LSR	53	1.38%	5.49%
LLP	52	1.19%	5.23%
LRR	60	0.97%	4.14%