Home assignment number 2, 2020, in SF2863 Systems Engineering

San Khaffaf[1] **and Alejandro Penacho**[2]

[1]**khaffaf@kth.se**
[2]**alejpr@kth.se**

MARGINAL ALLOCATION

1 ASSUMPTIONS FOR MALLOC

The number of spare parts of a line replacable unit, LRU, is denoted by x_j, where $j = 1, 2, ..., 9$. The vector $\mathbf{x}$ describes the number of spare parts of each LRU.
$f(\mathbf{x})$: expected number of grounded aircrafts at any moment, calculated as $f(\mathbf{x}) = EBO(\mathbf{x})$.
$g(\mathbf{x})$: total cost of the spare parts, calculated as $g(\mathbf{x}) = \sum_{j=1}^{9} c_j x_j$.

The problem then becomes to minimize the expected backorder, f, and minimize the total cost for spares, g, bounded by the C_{budget} = 500.

The required assumptions for f and g to be able to apply MALLOC, are that they are integer-convex, f is strictly decreasing and g is strictly increasing in each variable.

1.1 f decreasing and integer-convex

The following proofs that f is strictly decreasing and integer-convex apply to all variables. For f to be decreasing the following inequality has to apply:

$$\Delta f_j(x_j) \leq 0$$

If we let $R(x_j)$ denote the probability of shortage, the first inequality can be rewritten as below, proving that f is **decreasing** as the probability cannot be negative.

$$\Delta f_j(x_j) = EBO(x_j + 1) - EBO(x_j) = EBO(x_j) - R_j(x_j) - EBO(x_j) = -R_j(x_j) \leq 0$$

Integer-convexity for f requires that the inequality below applies.

$$\Delta f_j(x_j + 1) \geq \Delta f_j(x_j)$$

The inequality can be rewritten as:

$$\begin{aligned} \Delta f_j(x_j + 1) = EBO(x_j + 2) - EBO(x_j + 1) &= -R_j(x_j + 1) \\ \Delta f_j(x_j) = EBO(x_j + 1) - EBO(x_j) &= -R_j(x_j) \\ -R_j(x_j + 1) &\geq -R_j(x_j) \end{aligned}$$

As the probability of shortage will decrease with an increased number of spare parts, the inequality applies and f is **integer-convex**.

1.2 g increasing and integer-convex

As stated, g needs to be increasing and integer-convex for MALLOC to apply. The proofs below that g is strictly increasing and integer-convex apply to all variables. The following proves that g is an increasing function:

$$\Delta g_j(x_j) = g_j(x_j + 1) - g_j(x_j) = c_j x_j + c_j - c_j x_j = c_j \geq 0$$

The cost of all spare parts are positive, meaning this inequality is satisfied and g is **increasing**. Finally, as proven by the following, g is **integer-convex**.

$$\begin{aligned} \Delta g_j(x_j + 1) = c_j(x_j + 2) - c_j(x_j + 1) &= c_j \\ \Delta g_j(x_j) = c_j(x_j + 1) - c_j(x_j) &= c_j \\ c_j &\geq c_j \end{aligned}$$

2 ALGORITHM

At the start, there are no spare parts in the system for any of the line replaceable units. The values for EBO for this case can be determined since the intensity with which each LRU malfunctions and the average repair time for each LRU is known. The expected backorder for the case of no spare parts is calculated as:

$$f_j(x_j = 0) = EBO(x_j = 0) = \lambda_j T_j$$

The expected backorder will change for each added spare, and the values for it will depend on the change in probability of shortage. The probability of shortage can be computed for the case of zero spares and after that be computed recursively. For the case of zero spare parts, the probability of shortage for each variable becomes:

$$R_j(x_j = 0) = 1 - e^{\lambda_j T_j}$$

Following this, the probability of shortage and the EBO are determined using the following equations.

$$R_j(x_j + 1) = R_j - \frac{\lambda_j T_j}{j+1} e^{\lambda_j T_j}$$
$$f_j(x_j + 1) = EBO(x_j + 1) = EBO(x_j) - R_j(x_j + 1)$$

With these values determined, the marginal allocation algorithm can be performed. To start, the marginal cost of each LRU is computed for the case that one spare part is added to each LRU. The marginal cost is calculated as displayed below.

$$\frac{-\Delta f_j(x_j)}{\Delta g_j(x_j)}$$

A large marginal cost occurs when an added spare to LRU_j leads to a large decrease in EBO for a small increase in the cost. Therefore, the LRU that gives the largest marginal cost is chosen and a spare part of that unit is added if the cost of the spare part is small enough that adding it will not lead to going over the budget, otherwise the next largest one is chosen. Following this, the marginal cost for the chosen unit will now be determined for $x_j + 1$ and compared to the other marginal costs in order to repeat the process by finding the largest one. This process will be repeated until the cost is so large that spares cannot be added to any of the LRU's without surpassing the budget.
Each step in the process described above gives an efficient point of the solution. For each efficient point the expected backorder and the total cost is determined and the number of spare parts of each unit are noted.

3 RESULTS

The efficient solutions determined using the marginal allocation algorithm are listed below in Table 1. The table shows which line replaceable unit was provided with a spare in each step, as well as the cost, EBO and the marginal decrease in the EBO for each efficient point.

n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$x_5^{(n)}$	$x_6^{(n)}$	$x_7^{(n)}$	$x_8^{(n)}$	$x_9^{(n)}$	EBO	ΔEBO	$Cost$
0	0	0	0	0	0	0	0	0	0	4.379	-	0
1	0	0	0	0	1	0	0	0	0	4.158	0.221	10
2	0	0	1	0	1	0	0	0	0	3.690	0.468	35
3	0	0	1	1	1	0	0	0	0	3.465	0.225	50
4	1	0	1	1	1	0	0	0	0	3.284	0.181	64
5	1	0	1	1	1	1	0	0	0	2.706	0.578	109
6	1	1	1	1	1	1	0	0	0	2.461	0.245	128
7	1	1	1	1	1	1	1	0	0	1.698	0.763	208
8	1	1	1	1	1	1	1	1	0	1.454	0.244	241
9	1	1	1	1	1	1	1	1	1	1.289	0.165	271
10	1	1	2	1	1	1	1	1	1	1.157	0.132	296
11	1	1	2	1	1	1	2	1	1	0.736	0.421	376
12	1	1	2	1	1	2	2	1	1	0.521	0.215	421
13	1	1	2	1	2	2	2	1	1	0.495	0.026	431
14	1	1	2	2	2	2	2	1	1	0.467	0.028	446
15	1	2	2	2	2	2	2	1	1	0.435	0.032	465
16	2	2	2	2	2	2	2	1	1	0.417	0.018	479
17	2	2	2	2	3	2	2	1	1	0.415	0.002	489
18	2	2	2	2	4	2	2	1	1	0.415	0.0001	499

Table 1. Efficient solutions found using the marginal allocation algorithm

The efficient curve created by the solutions presented in Table 1 are displayed in the graph in figure 1.

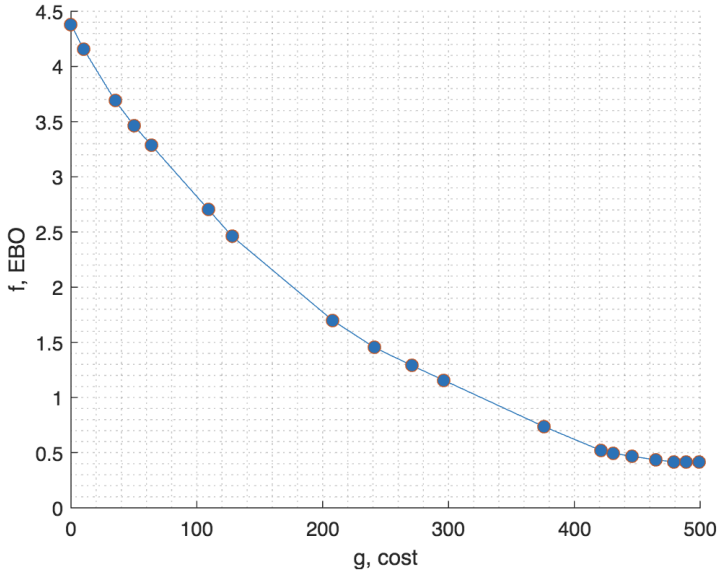

Figure 1. Efficient curve found with the marginal allocation algorithm

DYNAMIC PROGRAMMING

4

Stages

Each stage, refered as n, is defined as the budget already spent at that point, so stages go from 0 to 500. This is necessary in order to comply to the requirement of providing the optimal choice for any budget between 0 and the maximum one.

States

The state is given as a vector with the number of units of each *LRU* acquired. Although this definition massively increases the number of possible states of the system, it is has been found to be the only alternative possible. Since the variation of the *EBO* depends not only on the next *LRU* acquired but also on the number of those LRU already bought, any definition of state that does not carry the information regarding the number of *LRUs* is incomplete..

$$\mathbf{s}_n = \begin{bmatrix} s_n^1 & s_n^2 & s_n^3 & s_n^4 & s_n^5 & s_n^6 & s_n^7 & s_n^8 & s_n^9, \end{bmatrix}$$

where s_n^i defines the number of acquired LRU of type i at stage n.

State-update equations

At each state, the value of the decision variable x_n corresponds to the LRU to acquire next, so the stage and state evolve as follows:

$\Delta C = Cost(x_n)$	*The increase of cost is the price of the LRU acquired*
$n \longrightarrow n + c$	*The new stage to which the system jumps is given by the new total cost of the LRUs*
$\mathbf{s}_{n+c}^{x_n} = \mathbf{s}_n^{x_n} + 1$	*The number of LRU units of the type x_n increases by one*

Recursive relation

The system can reach a given stage in as many ways as *LRUs* are available. For example, stage 250 can be reached from stages 236 (buying LRU_1, $c = 14$), 231 (buying LRU_2, $c = 19$), etc. The optimal state of the system at stage 250 is that that provides the minimum *EBO*. As there are many possible states that correspond stages from which it is possible to arrive at 250, it is sensible to only consider those that have proven to be optimal. This process can be applied recursively starting from stage 0, and deriving optimal states for each stage until the last one, corresponding to the maximum budget, is reached.

Algorithm

The usual approach to dynamic programming problems is unsuitable to this problem due to how stages are defined. The usual procedure is to go stage by stage, obtaining the optimal path for each possible state at each one. In this system, the evolution of the system along them is not uniform. So, a slightly different approach was taken.

Previously, it was only necessary to consider one stage at each time, and compute all the paths connecting the states of this stage with the next one.

Instead, we will consider the ”cases”, that refer to a specific state of the system at an specific stage. Since the EBO is not dependant on the path taken to reach that point, only state and stage are necessary. At any given time, a set of active cases are considered, those which so far are considered to be optimal and are used to obtain the next active cases.

The programs starts by a root case, at stage 0. Then, the following procedure is followed:

1. The next optimal case is obtained by all the active cases taking the cases at the earliest stage, and taking the one with the smallest EBO. This ensures no better EBO for that stage will be found.
2. All the active cases with a higher or equal EBO are deleted, since they are not optimal. This is due to the fact that EBO always decreases with higher cost.
3. 9 new cases are generated by applying all possible decisions. That is, a case in which one units of LRU_1 is acquired, another in which an LRU_2 s acquired, etc.

5

After running the algorithm, a table showing the optimal array of *LRU* is obtained. It is important to note that there are not configurations for all possible costs. This is due to not only being that cost impossible to reach with the set of costs considered (if the cost is a relative prime of those of LRUs), but also because a lower EBO can be achieved with a lower cost. In that situation, all cases at that stage are considered suboptimal and rejected.

A reduced set of stages, with the optimal stage found for them, is shown in Table 2. As explained, there is now value for a cost of 150 because the optimal configuration found for 147 had a lower EBO.

Cost	LRU									EBO
	1	2	3	4	5	6	7	8	9	
0	0	0	0	0	0	0	0	0	0	4.379
99	0	1	1	0	1	1	0	0	0	2.868
147	0	1	1	1	1	1	0	1	0	2.398
346	1	1	2	1	1	1	2	1	0	0.9
498	1	2	2	2	2	2	2	2	1	0.402

Table 2. Strategies for different budgets

6

A new plot similar to Figure 1 is displayed, extending the results by also showing the points obtained using the dynamic programming approach. This approach shows a larger number of configurations, almost all of them inside the convex hull defined with the marginal allocation. While in general the results obtained with both methods are very similar, the dynamic programming approach is able to provide a more optimal final results for the budget of 500.

This is probably due to the fact that the the marginal allocation method, when only taking the optimal option at each step, limits the possibility of reaching a more optimal state that first requires taking a less optimal option. This seems to be specially limiting when approaching the limit budget. On the other hand, this approach is considerably less computationally demanding: it takes approximately 1/10 of the time of the dynamic programming method to arrive at the optimal point.

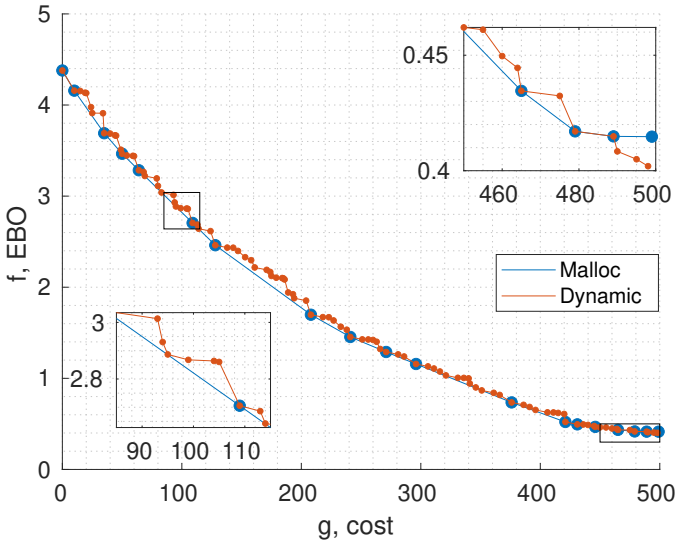

Figure 2. Optimal configurations obtained with malloc and dynamic programming