Finding the Optimal Stock Spring from Optimal Spring Design Characteristics

Manuel Paredes, Marc Sartor, Cédric Masclet

PAREDES M., SARTOR M., MASCLET C.

FINDING THE OPTIMAL STOCK SPRING FROM OPTIMAL SPRING DESIGN CHARACTERISTICS

https://doi.org/10.1007/978-94-015-9966-5_55

Abstract. This paper presents two methods for selecting the best stock spring from within a database. A way of calculating the optimal custom spring design which takes into account the database properties is described and two methods that explore the neighbourhood of the optimal continuous solution are detailed. Two methods for exploring the neighbourhood of the optimal continuous solution are discussed. The first, called the "expanding" method explores the database in successive layers, and stops once a solution has been found which cannot be improved upon. The second method also uses the "expanding" technique but stops once a spring which matches the specifications has been found. The algorithm then performs a "sliding" operator, whereby the immediate neighbourhood of the existing solution is explored. When no better spring can be found, this process stops. Two examples are shown. They demonstrate the effectiveness for our approach in selecting a stock spring without exploring the whole database. The most time-consuming stage is the calculation of the optimum custom-made spring design. Both these methods are particularly useful when applied to large database.

1. INTRODUCTION

The creation of mechanical objects is often the end result of a long design process where standard component selection is perhaps the simplest, but nonetheless frequent class of design decision problems. As catalogues become increasingly common and voluminous, this process can be time-consuming. Text-only assistance is not an efficient means for designers to find a stock spring which respects not only certain geometrical properties stored in a database but also several operating characteristics. Thus, optimal operating points have to be calculated for each potential stock spring depending on the requirements.

Finding the best stock spring and calculating its associated operating parameters is a Mixed Discrete Optimization problem. The most common mathematical solution is first to approximate an optimum by treating all variables as continuous and then to use algorithms to find the best feasible discrete point in the region of the continuous variable optimum. Optimization procedures dealing with continuous variables are numerous and well known. Technical literature provides mathematical methods which calculate design parameters corresponding to an optimal custom spring design (Sandgren (1990), Kannan and Kramer (1994), Deb and Goyal (1998)). To help designers select stock springs, Yuyi et al (1995) and Motz and Haghighi (1990)] have implemented methods that propose some stock springs close to the optimal

custom spring design. But, in every case, the designer has to calculate the operating parameters for each proposed spring by hand, in order to select the one that best fit his specifications. Moreover, as the best stock spring is not necessarily close to the custom-designed optimum, the designer is never sure he has selected the best spring.
To find the discrete optimum, many algorithms have been developed such as the optimal discrete search by Pappas and Allentuch (1974), the sequential linearization approach by Han Tong and Papalambros (1991) or the Boolean logic method by Peng and Siddall (1993). These methods explore the discrete neighbourhood of the continuous optimum but are difficult to implement. When there is a high number of discrete variables, they can however help to find the discrete optimum without enumerating the total neighbourhood of the continuous point.
Considering that spring catalogues give a maximum of four discrete variables in the optimization problem, our objective is to find methods that would extract the most suitable solution by just testing a few springs. Thus two methods have been developed. They use a particular optimal spring as a starting point to explore the database.

2. STARTING FROM A PARTICULAR OPTIMAL SPRING

Usually, optimal spring design is calculated by solving a conventional optimisation problem: Minimise or Maximise F($\boldsymbol{x}$) where $\boldsymbol{x}$ is a vector of continuous variables and F($\boldsymbol{x}$) is either the mass, or the fatigue life, or the operating load P2, or the operating length L2...
In addition to the objective function, a large set of constraints is considered, so as to express design specifications (operating length limits, operating load limits…), the standards (fatigue life, buckling length…) and the technical capability limits of the spring manufacturer (maximum outside diameter, minimum outside diameter, maximum wire diameter…).
But catalogues often provide a much smaller exploration domain than the hyper cube generally considered in the continuous optimisation process. The global optimum can therefore be a long way from the discrete approach. To match catalogue properties and decrease the gap between the two optima, constraints related to the database limits can be added to the initial optimisation problem. The augmented formulation will lead to a particular optimal spring which can be retained as an appropriate starting point for a catalogue search. The first task is thus to model the catalogue data in order to obtain the boundaries of the exploration domain.

2.1 Modelling the database

For this study, the catalogue of the «Ressort Vanel» company, comprising 5050 springs, has been used. For each spring, the characteristics detailed in the catalogue are stored in a table as shown in Figure 1. A spring reference can be for instance the index of the associated line in the table.

	Do	d	L0	R	n coils	Price
2371	5.3	0.5	32	0.38	16.8721141	
2372	5.3	0.5	40	0.3	20.8380112	
2373	5.3	0.5	50	0.24	25.547514	
2374	5.3	0.5	63	0.19	31.7442282	

Figure 1. Spring table

Table 1. Link between parameters

De	X	De	X	d	Y	d	Y	L0	Z
1,6	1	11	23	0,16	1	2	23	3,2	1
2	2	11,7	24	0,18	2	2,2	24	4	2
2,5	3	12,5	25	0,2	3	2,5	25	5	3
2,8	4	13,2	26	0,22	4	2,8	26	6,3	4
3,2	5	14	27	0,25	5	3,2	27	8	5
3,6	6	15	28	0,28	6	3,6	28	10	6
4	7	16	29	0,32	7	4	29	12,5	7
4,5	8	17	30	0,35	8	4,5	30	16	8
5	9	18	31	0,4	9	5	31	20	9
5,3	10	19	32	0,45	10	5,5	32	25	10
5,6	11	20	33	0,5	11	6	33	32	11
5,9	12	21	34	0,55	12			40	12
6,3	13	22	35	0,6	13			50	13
6,6	14	23,5	36	0,7	14			63	14
7	15	25	37	0,8	15			80	15
7,5	16	26,5	38	0,9	16			100	16
8	17	28	39	1	17			125	17
8,5	18	30	40	1,1	18			160	18
9	19	32	41	1,25	19			200	19
9,5	20	36	42	1,4	20				
10	21	40	43	1,6	21				
10,5	22	45	44	1,8	22				

Within this catalogue, springs are classified using three parameters : Do (outside diameter), d (wire diameter) and L0 (free length). Each spring can be represented by a point in the [Do, d, L0] space. Our suggestion is to add a matrix M[X, Y, Z] to this discrete space. The table 1 show the link between Do and X, d and Y, L0 and Z. Matrix dimensions are thus : M[44, 33, 19]. This matrix will contain only the spring references.
Note that M[] is not fulfilled by the database references. Some points of the discrete space have associated spring references, but not the others. For example M[10,11,11] = 2371 (see Fig 1) but M[1,33,19] = 0, as no spring exists with the dimensions Do = 1.6 mm, d = 6 mm, L0 = 200 mm.

2.2 Analysing the database limits

Figure 2 shows rectangles that surround the catalogue springs. The objective is to find the database limits in the three following planes : d/L0, Do/d and Do/L0. In each of these planes, a curve is sought that minimises the distance from the database (in number of cells within M[]) while staying outside the database limits.
Equations of the curves detailed in Figure 3 are:

$$dmax = 0.246\ Do^{0.9108},\ dmin = 0.07066\ Do^{0.9949}$$
$$L0max = 23.1\ Do^{0.6677},\ L0min = 1.4177\ Do^{0.8301}$$
$$L0max = 175.56\ d^{1.098},\ L0min = 0.07066\ d^{0.9049}$$

These equations are added to the initial optimisation problem in order to find an optimal custom spring design within the catalogue limits. In this study, $\boldsymbol{x}$ is defined with Do (outside diameter), d (wire diameter), L0 (free length), R (spring rate), L1 (maximum operating length), L2 (minimum operating length). From the result obtained, the values of Do, d, L0 are retained as starting point coordinates. Then, two exploration methods to select the most suitable spring from the database are presented.

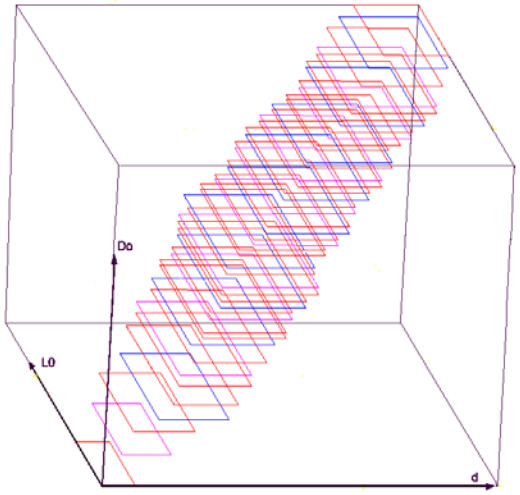

Figure 2. Do, d, L0 space

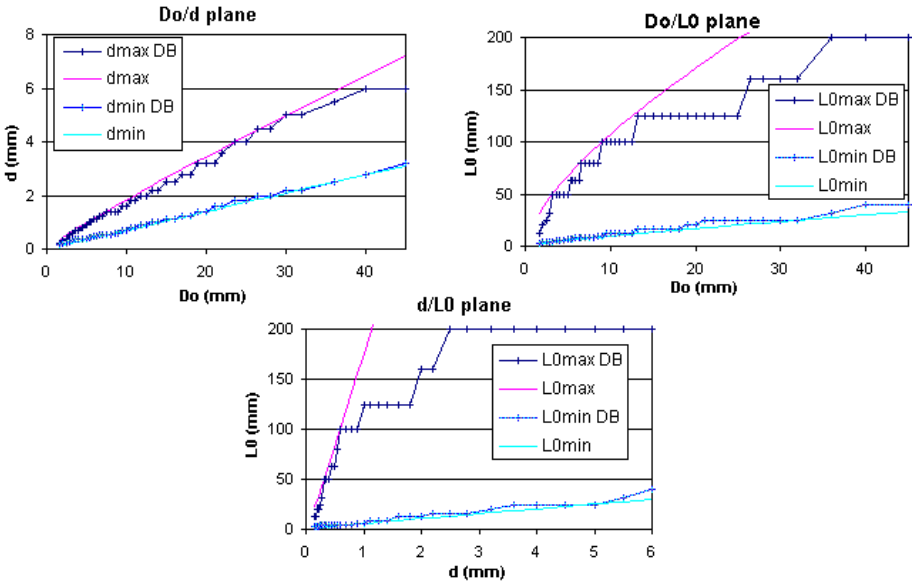

Figure 3. Limit curves

2. EXPLORATION METHODS

The optimal stock solution is not necessarily close to the custom-designed optimum. The following exploration methods test points on the matrix M[]. When a point is tested, the associated spring is evaluated in 5 steps (provided that the spring exists : M[X,Y,Z]≠0).

1 - Read the characteristics stored in the spring table
2 - Calculate the optimum operating points according to the specifications, the objective function and the standards (fatigue life, buckling length...) as described by Paredes (2000)
3 - Calculate the whole range of characteristics
4 - Calculate the objective function value
5 - Evaluate how the spring satisfies the requirements.

2.1 First exploration method : «Expanding»

The proposed method tests points that surround the starting point in successive layers, beginning with the points closest to the starting point (first layer), the others being gradually tested as described in Figure 4.

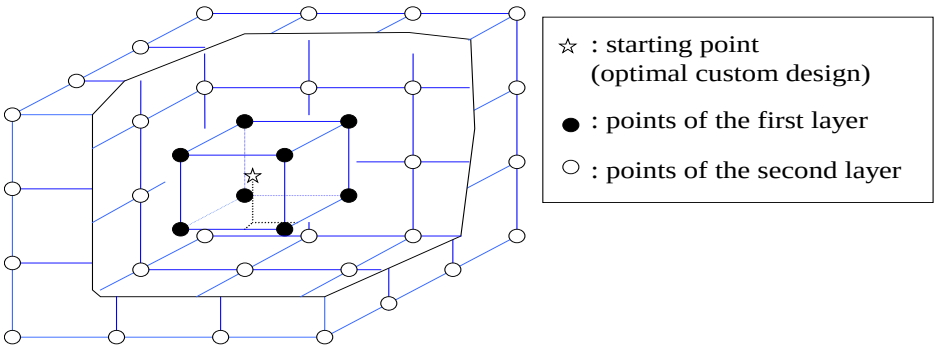

Figure 4. Expanding

In a n dimension space, a continuous point is surrounded by 2^n discrete points. In our case n=3, the first layer contains 8 points. The Nth layer contains the points located on the faces of a cube with sides 2N long. Thus, it contains $[6\ (2N-2)^2 + 24N - 16]$ points.
The progression is stopped when the exploration of a layer does not improve the solution obtained on the preceding layer. The stop criterion selected is thus as follows : the algorithm stops when, at the end of the exploration of the Nth layer, the optimal stock spring still belongs to the (N-1)th layer.

2.2 Second exploration method : "Expanding, then sliding a window"

When the dimensions of the database spatial model are large (n>2), the exploration of numerous layers results in the testing of a large number of points. An alternative approach can then be used : "Expanding, then sliding a window". The exploration starts as defined previously but when at least one spring matching the specifications has been found on a layer, the expansion is stopped and the process continues by sliding a window.

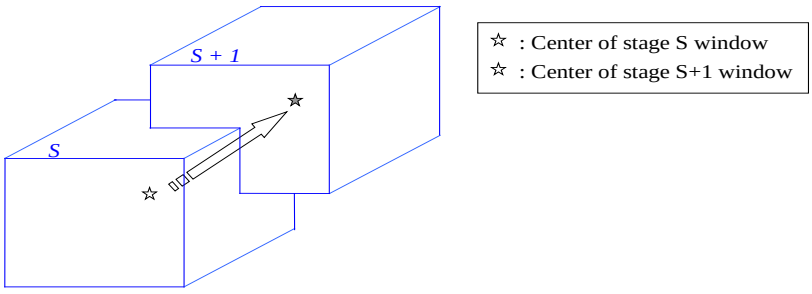

Figure 5. Sliding a window

At this stage, the algorithm only tests the points from within a window that surrounds the optimal point previously found as shown in Figure 5 (discrete neighbourhood). For an n dimension space and a R window radius, a discrete point is surrounded by $(1+2R)^n - 1$ points. In our case, the unit-neighbourhood is tested inducing n = 3 and R = 1 : 26 points are thus evaluated. If a better point is found, the window is slid onto that new point. The algorithm is stopped when the stage has not improved the preceding solution.

3. EXAMPLES

Two examples are given below to illustrate and compare the efficiency of the two methods. Note that, all the springs in the database have been simultaneously evaluated in order to control the results. Both these examples deal with, steel springs with closed and ground ends.

3.1 Minimum mass

In this example, the spring with the smallest mass value and a spring travel of 10 mm, a minimum inside diameter (Di) of 4 mm, F1 equal to 15 N and a maximum L1 of 35 mm is sought. The optimal custom spring design within the database limits induces Do = 6.42 mm, d = 0.75 mm, L0 = 39.69 mm. Thus, according to table 1, the first layer cube coordinates are [13, 14, 11], [14, 15, 12] (points of the diagonal). Results obtained with both methods are shown in Table 2.

Table 2. Minimum mass

Method	Expanding	Exp. then sliding
Nb of layers	5	2
Nb of windows	-	3
Nb of springs	656	**142**
Result coordinates	17, 15, 13	17, 15, 13

The algorithm proposes in every case [17, 15, 13] as a solution, i.e.
Do = 8.0 mm, d = 0.8 mm, L0 = 50 mm, R = 0.68 N/mm, working between L1 = 27.94 mm and L2 = 17.94 mm.
Checking procedure: the evaluation of the entire database (5050 springs) leads to the same stock spring (298 springs match the specifications). In this example, the best method is "Expanding, then sliding a window". In this case, the optimal stock spring can be found after evaluating only 142 springs.

3.2 Minimum operating length

Prototype design specifications are : Do ≤ 38mm, Di ≥ 27mm, L1 ≤ 50mm, spring travel equal to 11mm. 5N ≤ P1 ≤ 15N and 50N ≤ P2 ≤ 100N (spring loads). The spring with the smallest operating length is sought. Using the optimal design in continuous variables obtained without adding the database constraints induces first layer coordinates of : [40, 22, 10], [41, 23, 11]. When these constraints are added to the optimization problem, the first layer coordinates become [40, 23, 9], [41, 24, 10]. The results obtained are detailed in Table 3.

Table 3. Minimum operating length

Starting point	without adding constraints		when adding constraints	
Method	Expanding	Exp. Then sliding	Expanding	Exp. then sliding
Nb of layers	3	2	2	1
Nb of windows	-	1	-	1
Nb of springs	43	**20**	15	**10**
Result coordinates	41, 24, 10	41, 24, 10	41, 24, 10	41, 24, 10

In every case, [41, 24, 10] is proposed as a solution, i.e. Do = 32 mm, d = 2.2 mm, L0 = 25 mm, R = 5.78 N/mm, working between L1 = 22.4 mm and L2 = 11.4 mm.
Checking procedure : the evaluation of the entire database leads to the same stock spring from the 12 springs that match the specifications. Once again, it shows the efficiency of the "Expanding, then sliding a window" method. It also shows the importance of adding the database constraints to the initial optimization problem in order to reduce the number of springs evaluated.

4. CONCLUSION

This paper presents two methods for selecting stock springs from within a catalogue. These methods are less time consuming than that which tests all the springs, especially for large catalogues. The first method consists in testing the springs that surround the optimal custom spring design. The algorithm called "Expanding" tests springs by successive layers, moving away gradually until the exploration of a layer no longer improves the solution selected on the preceding layer. The second method starts with a similar expansion process and stops as soon as a spring that satisfies the specifications is found. Then the springs from within a window that surrounds that spring are tested. If a better spring is found, the window is slid onto this point and a new step is carried out until no better solution remains. The results obtained are satisfactory and have a high "result" VS "number of tested springs" ratio. This ratio can be increased by adding the database limit constraints to the initial optimization problem of finding the optimal custom spring design. As it reduces calculation time, this method is particularly useful for large data-bases. It can also help in selecting a temporary stock spring to replace a broken custom-made spring, as here the starting characteristics are already known. In addition, it is particularly important to reduce operating time in integrated design procedures.

5. AFFILIATION

PAREDES Manuel, SARTOR Marc, MASCLET Cédric
LGMT - INSA, 135 avenue de Rangueil 31077 Toulouse (France)
Tel : 33 5 61 55 97 18 Fax : 33 5 61 55 97 00
email: manuel.paredes@insa-tlse.fr
web: http://www.meca.insa-tlse.fr/lgmt

6. REFERENCES

Deb, K. & Goyal, M. (1998). A Flexible Optimisation Procedure for Mechanical Component Design Based on Genetic Adaptive Search. *Journal of Mechanical Design*, *120*, 162-164.

Han Tong, L. & Papalambros, P.Y. (1991). Computational Implementation and Test of a Sequential Linearization Algorithm for Mixed-discrete Nonlinear Design Optimization. *Journal of Mechanical Design*, *113*, 335-345.

Kannan, B. K. & Kramer, S. N. (1994). An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimisation and its Applications to Mechanical Design. *Journal of Mechanical Design*, *116*, 405-411.

Motz, D. S. & Haghighi, K. (1990). An Integrated Approach to Knowledge-Aided Design and Optimisation of Mechanical Springs. *Transactions of the ASAE, 33(5)*, 1729-1735.

Pappas, M. & Allentuch, A. (1974). Mathematical Programming Procedures for Mixed-Continuous Design Problems. *Journal of engineering for Industry*, 201-209.

Paredes, M., Sartor, M. & Fauroux J.C. (2000). Stock Spring Selection Tool. *SPRINGS*, *winter*, 53-67.

Peng, L. & Siddall, J.N.(1993). A Boolean Local Improvement Method for General Discrete Optimization Problems. *Journal of Mechanical Design*, *115*, 776-783.

Sandgren, E.(1990). Nonlinear Integer and Discrete Programming in Mechanical Design Optimisation. *Journal of Mechanical Design*, *112*, 223-229.

Yuyi, L., Kok-Keong, T. & Liangxi, W. (1995). Application of Expert System for Spring Design and Procurement. *SPRINGS*, *march*, 66-80.