CHAPTER 9

DESIGN STUDY TOOL ALGORITHM

This chapter discusses the assembly design tool algorithm and methodology. The algorithm described in this chapter is used to search the design space as well as post-process the results to aid the decision making process in assembly design. The design tool module has three kinds of capabilities: constraint modification, constraint reduction, and constraint addition. Among the three, constraint modification is the one that is computationally most intensive. This is due to the many permutations in searching the specified design space. Without an efficient algorithm, the computation time can be impractical.

9.1 Constraint modification

The goal in constraint modification is to observe the change in assembly rating as the location and orientation of constraints are modified. Similar core algorithms based on the analysis tool are used. The essential functions of the analysis tool are to combine pivot constraint combinations, generate the motion, and evaluate the resistance of reaction constraints to that motion. The design tool core functions in exploring the design space are to define the design search space, sort out recalculations involving the original motion set, generate a new motion set and the respective resistance values, and summarize the results in a response surface plot.

9.1.1 Types of design search space and optimization variable specification

The location of a constraint can be varied at discrete locations, along a straight line, along a circular curve, and across a plane. The orientation of a constraint can be varied in two different directions. The size of a constraint can also be varied. Not all of these search spaces are applicable to all constraints. For example, the size search space is not applicable to point and pin constraints for obvious reasons. The design space that is applicable for each constraint type is shown in Table 9.1.

	Location Search Space				Orientation search space		Size search space	
Constraint Type	**Discrete**	**Line search**	**Curved line search**	**Plane**	**Normal direction**	**Line direction**	**Length**	**Area**
Point	x	x	x	x	x			
Pin	x	x	x	x	x			
Line	x	x		x		x	x	
Plane	x	x		x	x			x
Search dimension	1D	1D	1D	2D	1D or 2D	1D or 2D	1D	1D
Revision Type	1	2	3	4	5, 6	7	8	9,10

Table 9.1 Design space mapping for each constraint type

Since assembly features are usually represented by multiple constraints, even after HOC modeling is utilized, the constraints need to be modified in groups. Optimization variables are typically specified as a variable group. Grouping of constraints allows multiple constraints in one group to be simultaneously relocated, reoriented, or resized. At the beginning of each case study, these variable groups and the search space are defined in a table. The search spaces are defined in the following parameters:

- The discrete search space is a set of candidate point locations for the constraint variable. The number of constraint variables for this kind of search space is limited to one.
- The line search space (Figure 9.1a) is defined by its center point L_o, its direction vector $\vec{L}_d$, and the move limit in either direction δ.
- The circular curve search space (Figure 9.1b) is defined by its center point L_o, its rotation axis $\vec{L}_R$, and the angular move limit in either direction θ.
- The plane search space (Figure 9.1c) is defined by its center point P_o, its two principal direction vectors $\vec{P}_x$ and $\vec{P}_y$, and the move limit in either direction for both axes δ_x and δ_y.
- The normal orientation search space can be defined as rotation around one or two axes, depending on the dimension of the search of interest. The search space is defined by its rotation axes $\vec{r}_1$ and $\vec{r}_2$ and the angular move limit in either direction θ_1 and θ_2.

- The line direction orientation search space is defined by its rotation axis $\vec{r}$ and the angular move limit θ in either direction.
- The line size search space is defined by its size limits l_l and l_u.
- The plane area size search space is defined by its size limits in its two principal directions, l_l, l_u, w_l, and w_u in the case of rectangular plane constraint and r_l and r_u in the case of circular plane constraint.

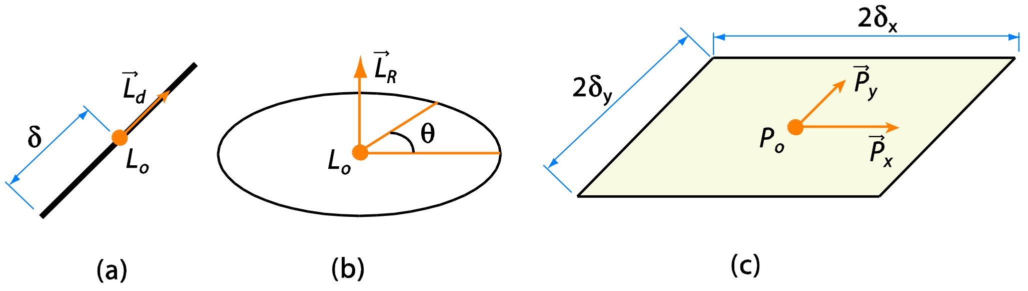

Figure 9.1 Illustration of the (a) line, (b) curved line, and (c) plane search space

The MATLAB script developed in this dissertation is capable of searching the search spaces listed above, but not all of these search spaces are used in the case studies.

9.1.2 Factorial search methodology in design space exploration

In order to understand the effect of different constraint strategies on assembly characteristics, the design space needs to be well understood. Therefore, instead of solely identifying the optimum point in the optimization search, the whole design space is explored and presented in a response surface plot.

The dimension D in the optimization process is determined by the number of constraint variable groups n and the dimension of the search space S for each constraint variable group i. The dimension is calculated by

$$D = \sum_{i=1}^{n} S_i. \tag{9.1}$$

A full factorial search is implemented in the MATLAB script as D number of nested for-loops.

The variable x is the increment along the search dimension. In the case of multiple dimension searches, x is an array of length D. Each search dimension limits are normalized to $-1 \leq x \leq 1$, with -1 and 1 corresponding to the minimum and maximum limits of each search dimension. In the case studies, the number of increments is typically set to 10 or less to maintain a realistic computation time while collective enough data points to plot the optimization result.

9.1.3 Constraint modification pre-processing

It is important to understand that in the constraint modification scheme, the evaluated motion set for each constraint modification iteration is unique. The methodology of assembly constraint analysis is recursive in some ways. The motions to be evaluated are generated from the constraint configuration itself. Therefore, as the constraint configuration changes, so do the evaluated motions.

When constraints are modified, there are two types of recalculations involved in the process. One is the recalculation involved when modified constraints act as reaction constraints. The other is when they act as pivot constraints. The recalculated motion in the first case is called recalculating the resistance for ‘base motion’ (Section 9.1.5) because the motion specifications do not change, but the modified constraint resistance values need to be recalculated. The recalculation in the second case involves generating the ‘new motion set’ (Section 9.1.6) because the motion specifications change. The motion changes because the modified constraints are members of the pivot constraint set.

The preprocessing of the constraint modification algorithm involves separating these two types of motion. Each motion is identified with the pivot constraint set to which it is reciprocal. The following procedure is done to separate the motions:

1. The constraints that are subject to be modified from all optimization variable groups are collected as a set.
2. Pivot constraint combinations that contain any of the modified constraints are identified. From these combinations, the ‘new motions’ are to be generated and recalculated.
3. The motions associated with the pivot constraint combination identified in step #2 are to be removed from the original rating matrix. However, exceptions apply to motions that are duplicates of motions associated with other pivot constraint combinations. Because they are associated to multiple pivot constraint combinations (including those that do not contain any to-be-modified constraints), they should not be removed.
4. After these exceptions are identified, the combinations identified in #2 are removed from the original rating matrix. The motions that remain are identified as the ‘base motion’ set.

5. The resistance of the modified constraints to this ‘base motion’ set will be recalculated and will replace the old resistance values.

This pre-processing is only done once before proceeding to the nested for-loop that searches the specified design space.

9.1.4 Incremental constraint modification

Each iteration in the nested for-loop begins with modifying the constraint to the next increment in the search space. For the details of this constraint, please refer to the comments in the MATLAB script for each search type attached in Appendix A.

9.1.5 Recalculation for base motion set

The resistance of the modified constraints to the base motion set needs to be recalculated. The subroutine that runs this task is similar to the main processor from the analysis tool. In Equation 7.22, the pivot constraint wrench $\vec{C}_{P1,P5}$ can be composed from the original pivot constraint combination in most cases. However, when the pivot constraint combination contains one of the modified constraints, the pivot constraint set is no longer reciprocal to the evaluated motion. These refer to the exception identified in step #3 from Section 9.1.3. In this case, the pivot constraint wrench needs to be adjusted in order to be either of rank 5 or to be replaced by an alternate pivot constraint that is also reciprocal to the motion. For the details of this adjustment, please refer to the comments in the MATLAB script attached in Appendix A. This subroutine is coded in the script *rate_motset.m.*

9.1.6 Recalculation for new motion set

The subroutine that runs this task processes the pivot constraint combinations identified in step #2 from Section 9.1.3. The main processor from the analysis tool described in Section 7.3 is used for this task (*main_loop.m*).

9.1.7 Constraint modification post-processing and optimization plots

The recalculation of the 'base motion' set as well as the 'new motion' set is done for each incremental constraint modification. A new rating matrix is created by merging the results from the recalculation procedures. The overall assembly rating is calculated from the rating matrix and stored for each incremental search of the design space.

Based on the stored results, the change in rating from the baseline rating is calculated. The change in assembly rating is reported as a response surface plot. For one- to two- dimensional optimization searches, a response surface plot is displayed to give design space information to aid design decisions. The response surface plot is useful in understanding the design space; therefore, in most case studies the optimization search is limited to two dimensions. For optimization searches of more than two dimensions, only the optimum point is identified and reported.

9.1.8 Constraint modification flowchart

The overall flowchart for the constraint modification design tool is shown in Figure 9.2.

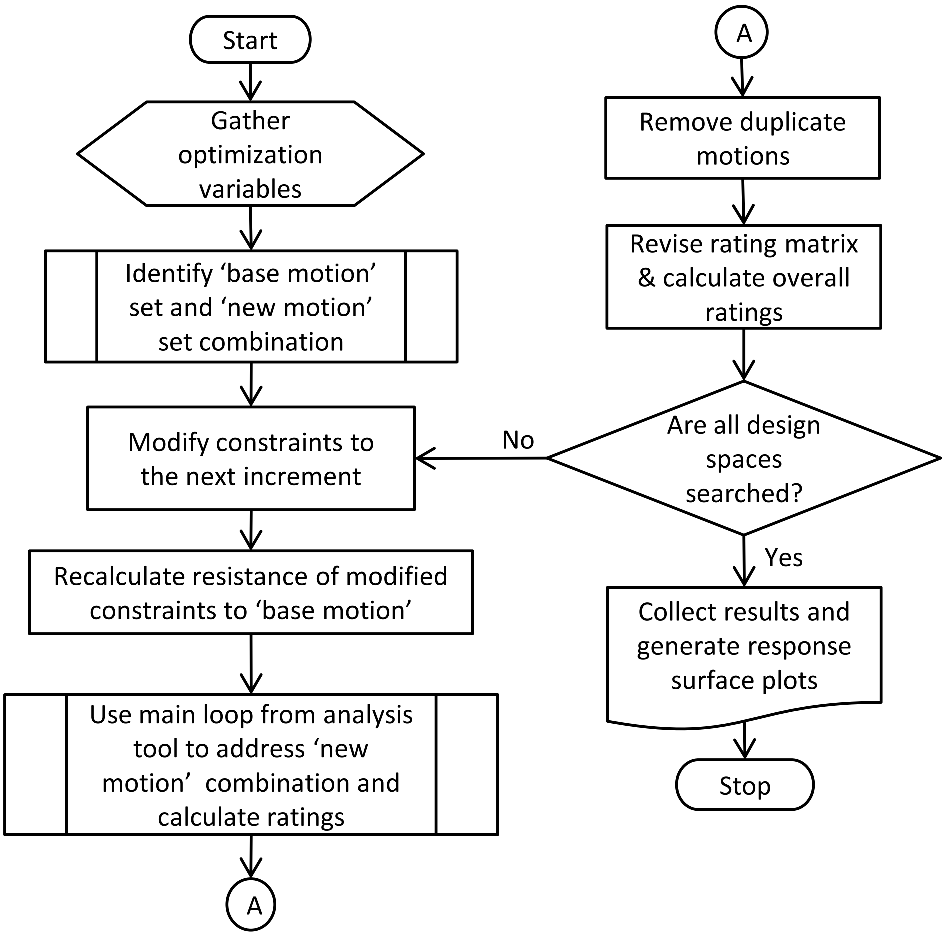
Start
Gather optimization variables
Identify ‘base motion’ set and ‘new motion’ set combination
Modify constraints to the next increment
Recalculate resistance of modified constraints to ‘base motion’
Use main loop from analysis tool to address ‘new motion’ combination and calculate ratings
A
A
Remove duplicate motions
Revise rating matrix & calculate overall ratings
Are all design spaces searched?
No
Yes
Collect results and generate response surface plots
Stop

Figure 9.2 Simplified flowchart for constraint modification algorithm

9.2 Constraint reduction

The goal of the constraint reduction algorithm is to observe the change in assembly ratings as constraints are removed. Similar to the constraint modification algorithm, the goal is to observe and understand the design space. In this case, the design space is composed of the number of constraints to remove and which constraints to remove.

The number of constraints to remove is specified by the user. The MATLAB script is programmed to conduct the optimization search based on the number of constraints. It is also possible to program the script to conduct the optimization search based on the percent redundancy ratio decrease or TOR increase; however, this capability is not implemented due to the goal of exploring the design space.

If the designer wishes to remove one constraint, there are n different constraints that one can remove. Instead of using certain criteria to select the best candidate, the algorithm searches through all cases of constraint reduction and displays the assembly rating change due to each case of constraint removal. In the case where the designer wishes to remove k constraints, there are $\frac{n!}{k!(n-k)!}$ different ways to remove k constraints at a time. The assembly rating change due to each combination of constraint removal is displayed.

Because the purpose of constraint reduction is to reduce redundancy while achieving the least decrease in the assembly's capacity to resist motion, the TOR is of interest. Based on the optimization search results described in the paragraph above, the constraint reduction removal case that yields the maximum TOR increase is selected to be the best.

9.2.1 Constraint reduction algorithm

The constraint reduction algorithm is very similar to the constraint modification preprocessor (Section 9.1.3). The constraint modification preprocessor identifies the motion where the reciprocal pivot constraint contains a modified constraint and eliminates them with the same exception identified in Section 9.1.3. The constraint reduction algorithm accomplishes a very similar task. It needs to eliminate motions where the pivot constraint contains the removed constraints. In addition to the motion elimination, the constraint reduction algorithm also removes the column in the rating matrix that is associated with the currently removed constraint. The modified rating matrix is then evaluated for the revised assembly rating, and the difference from the original assembly rating is calculated.

This process is repeated for each constraint (in the case of one-at-a-time constraint removal) or for each constraint combination (in the case of more than one at a time constraint removal). The result of the constraint reduction algorithm is displayed as a line plot where the horizontal axis refers to the constraint index. Each index value is associated with a constraint or multiple constraints being removed at a time. The multiple constraint removal combination that yields the most TOR increase is also reported.

9.3 Constraint addition

The goal of the constraint addition algorithm is to observe the change in assembly rating as constraints are added. Added constraints result in an increase in the total resistance rating as well as an increase in redundancy. This algorithm along with the constraint reduction algorithm is to be used in the trade-off case studies.

The input file specifies the type of constraint being added and the constraint itself. The MATLAB script adds these constraints one at a time. In the future, this can be improved by adding the capability to add multiple constraints at a time to identify the combination that will yield the most gain in MTR.

9.3.1 Constraint addition algorithm

The constraint addition algorithm is fairly simple. There are two additional calculations that need to be done once a new constraint is added. First, the resistance values for the original motion need to be calculated. Second, new pivot constraint combinations that contain the newly added constraint need to be processed. The first calculation uses the same subroutine as described in Section 9.1.5, and the second calculation uses the main processor from the analysis tool. The revised assembly rating for each added constraint is reported in the result.

9.4 Specified loading condition optimization

The goal of the specified loading condition optimization is to observe the change in total resistance rating for a specific loading condition. This optimization search modifies the constraint based on the same parameter as the constraint modification optimization (Section 9.1), but evaluates only the motion that is specified as the loading condition of interest. The evaluated motion set in this case is specified by the user and not generated from a set of pivot constraint wrenches. Because of this, the vectors $\vec{C}_{P1,P5}$ in the static equilibrium equation 7.22 are not formed from a pivot wrench set. Instead, the null space of the specified motion is calculated and used as $\vec{C}_{P1,P5}$. This five-system is treated as a pseudo-pivot constraint wrench set for completing the equilibrium equation.

9.5 Sensitivity analysis study

The goal of the sensitivity analysis study is to observe the change in assembly ratings when constraints are perturbed. Due to manufacturing variation and defects, constraints might either be deactivated (not in contact) or vary from their nominal location and orientation. By perturbing some of these constraints and observing the effects on the ratings, a sensitivity analysis can be done to evaluate assembly design robustness with respect to dimensional variation. To simplify the study at this stage, only the WTR rating is observed. There are three kinds of sensitivity analysis algorithm implemented in this study, namely toggle perturbation, position perturbation, and orientation perturbation. The results are reported as the percent change in WTR for each constraint that is perturbed.

The toggle perturbation algorithm deactivates one constraint at a time and evaluates the change in WTR rating. This algorithm uses the same function from constraint reduction with constraint removal specified as one at a time (Section 9.2.1).

The position perturbation algorithm perturbs the location of one constraint at a time and evaluates the change in assembly rating. This algorithm uses the same function from the constraint modification with plane search space optimization (Section 9.1.1). The amount of position perturbation is specified by the user.

The orientation perturbation algorithm perturbs the orientation of one constraint at a time and evaluates the change in assembly rating. This algorithm uses the same function from the constraint modification with orientation search space optimization in two angles (Section 9.1.1).

9.6 Summary

This chapter discussed the algorithm and methodology implemented in the design tool to accomplish the objectives explained in Chapter 8. Chapters 10 and 11 will discuss the case studies that use the analysis tool and the design tool