INTERNALTIONAL CONFERENCE ON ENGEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15 – 18, 2005

THREE-DIMENTIONAL DESIGN LAYOUT OPTIMIZATION FOR MECHATRONIC DEVICES USING THE SIMULATED ANNEALING APPROCH

Hao Shen, Aleksandar Subic, Anna Bourmistrov

Abstract:

In this paper we look at a combinatorial optimization problem in application to a layout of mechatronic devices. There is a need to create a flexible and intelligent design tool, which allows an engineer to handle the complexity and non-linearity of a 3D mechatronic design task. The objective of the research is to develop a feasible optimization algorithm based on Simulated Annealing (SA). SA is a family of randomized algorithms used to solve many combinatorial optimization problems. This paper presents a conceptual model of integrated design environment based on SA algorithm embedded into Solid Works. The focus is on structural optimization with housing and layout modeling, with the objective to minimize the required assembly space. Also we present an attempt to automate the design process by introduction of integrative modeling of a set of design modules, which can be selected and linked together to provide tailored modeling solutions to a variety of design problems.

Key Words: Simulated Annealing, 3D Modeling, Design Simulation and Concurrent Optimization.

1. Introduction

Space optimization for an assembly of components in complex hybrid systems is a challenging structural design task. Experience and common sense have been usual engineering tools in the optimization process. Although CAD environment helps designers to achieve compact designs to a certain degree, it is still time consuming and the solution is less than optimal.

Modeling and simulation techniques allow flexible and optimal handling of a complex *n* dimensional design space. Thus it becomes essential to create an intelligent design environment which gives the benefit of SA optimization in loop with the comprehensive Solid Works model. Development of capable optimization tool for structural design is a contribution towards improving a feasibility of engineering efforts.

SA [1, 2, 3, 4] mimics the physical annealing process and is successfully used to obtain an intelligent solution for a variety of combinatorial optimization problems. The process can be formulated as the problem of finding – among a potentially very large number of solutions – a layout solution with lowest energy or cost function. In other words, an example of a combinational optimization problem can be formulated as a pair *(R, C),* where *R* is the finite or possibly countable infinite set of *configurations* and *C* is a *cost function*. It is assumed that *C* is defined such that the lower the value of *C*, the better the corresponding configuration, with respect to the optimization criteria.

During the past decades, SA technique has been utilized in such diverse areas as, for example, electrical engineering, nesting problem, packing problem, operation research, code design, image processing, and molecular physics. In many of these areas the exiting algorithms performed poorly whilst SA has shown such salient features as general applicability, flexibility, ease of implementation and a modicum of sophistication. Embedding annealing optimization procedure into a CAD system will be an exciting challenge to achieve an intelligent design environment of significant potential for a wide range of possible engineering applications.

This paper presents the concept of space optimization methodology for mechatronic devices within a 3D envelope. Design requirements and constraints are based on a number of different components and interfaces within a device. The embedded optimization procedure intelligently creates an optimal solution for a minimized, highly integrated assembly. This optimized solution is then fully designed in Solid Works CAD environment using the identified envelope as the design space.

2. Related Research Work

SA is a general method for a wide variety of combinatorial problems. The method is flexible as well as powerful.

Simulated Annealing algorithms have shown great success in the circuit layout design and significant work in the mechanical component layout area has been motivated by the circuit layout technology. The performance of the SA algorithm has been compared to other facility layout methods and shown to yield either equal or better quality for each of classical test problems. The efficiency of the SA approach is recognized to be insensitive to initial starting states.

Szykman [5] and Cagan [6] extended the technology from *2-D VLSI to 3-D mechanical and electro-mechanical layout.* Their approach can deal with blocks and cylinders with rotations constrained to multiples of 90°. A perturbation-based approach was used in which infeasible states with component overlap and constraint violations were allowed and penalized.

An integrated SA approach to 3-D layout and routing was introduced in Szykman and Cagan [7] and Szykman et al [8], and the experiments showed that concurrent layout and routing results in superior solutions over the traditional layout–then–rout approach. Routing problems are abundant in engineering applications such as routing of pipes, wires and air ducts. The routing cost can be influential in the manufacturing of certain products such as HVAC (Heat, Ventilation and Air Conditioning) products. Taking the routing cost into account during the component placement stage of the layout could significantly improve the quality and reduce the cost of the product layout.

Rao and Iyengar [9] applied SA to a variety of the bin-packing problems. The extensive simulation experiments demonstrated that the solutions obtained by SA showed a significant improvement over those obtained by any other well-known heuristic methods. Han and Na [10] proposed a nesting approach with two stages: initial layout stage and layout improvement stage. A self-organizing layout algorithm generates a 'good' initial layout then Simulated Annealing was used to improve the initial layout. In short, it has been demonstrated that the Simulated Annealing is capable of successfully solving bin-packing problems. However, in all applications the performance, in terms of efficiency and effectiveness, depends greatly on the solution space, implementation strategies and objective functions.

3. Simulated Annealing

SA is a generally applicable, stochastic technique based on the analogy between simulating the metallurgical annealing process and solving large combinational optimization problems. The algorithm was first derived by Metroplis et al. [11] in 1953 then was further extended by Kirkaptrick et al. [12] in 1983.

SA is a kind of hill-climbing search for finding a good solution. A simple hill-climber searches in the neighborhood of the best solution found to date, and jumps to a new solution whenever one is found that improves upon the best to date. Formally, Simulated Annealing is a *Markov Chain Monte Carlo* method. That is, the probability of jumping from one point to another depends only on the last point and not on the entire previous history.

Within the algorithm, an initial design state is chosen and the value of the *cost function* for that state is evaluated. A step is taken to a new state by applying a *move*, or operator, from an available *move set*. This new state is evaluated; if the step leads to an improvement in the cost function, the new design is accepted and becomes the current design state. If the step leads to an inferior state, the step may still be accepted with Boltzmann probability. This probability is a function of a decreasing parameter called *temperature,* based on an analogy with the annealing of metals, given by:

$$Probability_{\text{accept}} = e^{-\Delta C/\mathrm{T}} \qquad (1)$$

Where ΔC is the change in cost function due to the move and T is the current *temperature*. The $Probability_{\text{accept}}$ would be a uniform random number between 0 and 1 ($0 < Probability_{\text{accept}} \leq 1$).

Applications of SA algorithms require specifications of three distinct items: (1) a concise problem representation- (2) a transition mechanism; and (3) a cooling schedule. The problem representation consists of a configuration representation and an expression for the cost function. The *cost function* represents the cost effectiveness of different layouts. The transition mechanism generates a new configuration from a current one. The difference in cost between the two configurations must be calculated and a decision is made whether or not the new configuration is to be accepted. The cooling schedule is used to control the *temperature* in the algorithm, and specify the starting value, decrement function, length of generation and the stopping criterion.

The *temperature* starts out high and decreases with time. At each *temperature*, the system is perturbed several times. The set of iterations carried out at each value of the *temperature* is called a *Markov Chain.* The number of iterations in a chain is sometimes referred to as the *chain length.* Initially, steps taken through the state space (and, therefore, the cost function space) are almost random, resulting in a broad exploration of the cost function space. As the probability of accepting inferior steps decreases, those steps tend to get rejected, allowing the algorithm to converge to an optimum once promising areas of the cost function space have been found. And, as the temperature freezes, Simulated Annealing completes its search like a simple hill climber.

The process has the objective to find an optimal solution with the minimal associated cost. Sorkin [13] has revealed that the behavior of simulated annealing depends heavily on the cost function landscape associated with the optimization problem. The success of annealing relies on the overall ‘cost difference’ of collections of states being large compared with the barriers dividing these collections. A large number of evaluations are necessary for SA to converge to good solutions. When the cost function is complex, the computation can be expensive and time-consuming.

4. Three-Dimensional Design Layout Optimization

Space optimization for electric-mechanical assemblies is a time-consuming and challenging task for designers. Although CAD environment can help designers to achieve compact designs to a certain degree, the solutions are still less than optimal. Embedding annealing optimization procedure into CAD system is an exciting challenge aiming to achieve an intelligent design environment of significant potential for a wide range of applications. The undertaken research is a new trial of using SA technique embedded in SolidWorks environment to optimize complex mechatronic assemblies. Such space optimization method can be used as an optimization tool to intelligently minimize the housing space for many complex devices where a number of different sections and components are interfaced in the assembly.

4.1 Problem formulation and solution strategy

In this work, the 3D layout problem can be described as packing a batch of components of different sizes into an ‘envelope’. The packing tasks are characterized by the following four objectives:

- Fitting the components into the specified envelope;
- Observing topological connections or assembly relationships between components;
- Avoiding any overlap between components and the envelope protrusion;
- Achieving high packing density, or minimizing the overall space of the envelope.

Obviously, this layout problem is non-liner, combinational and discontinuous. That makes SA algorithm a promising approach in finding the global, or at least very good, local optimum. The strategy is to *move* components around within a pre-defined space and analyze each *move* on its effectives towards minimizing a combined cost function. The function includes design goals such as minimizing the packing density, the relationships between components, and the penalty terms, that evaluate the amount of components overlaps, envelope penetrations and spatial constraint violations.

The *move set* can be defined as following:

- *Rotate* – to change the orientation of a component without changing the location of the component center. There are six possible orientations to be selected.
- *Stroll* – to randomly change the location of the center of the component in ±x, ±y, ±z directions. The moving distance depends on the current temperature. A longer distance

of movement will be selected at a higher temperature to achieve a greater change in the cost function and to avoid local minimum.

- *Swap* – to switch the locations of two components. The two components are selected randomly.
- *Move towards the origin (0, 0, 0)* – to move a randomly selected component along with the x, y, z coordinate directions of the packing envelope, so that the overall envelope could be decreased accordingly.
- *Move against wall* – to move a randomly selected component towards the nearest wall of the packing envelope until it touches other component or the wall, and then reduce the packing envelope.
- *Eliminate overlap* – to calculate the overlap vector between components then move each component along the corresponding vector towards decreasing the overall overlaps.

Upon the nature of Simulated Annealing, large numbers of evaluations must be performed throughout the hundreds of thousands of moves. Therefore, the evaluation of the cost function must be performed quickly and effectively. The detection of overlaps and its quantification is the most computationally expensive task. Thus, to minimize the overall running time, the algorithm must minimize the complexity within this operation.

4.2 Overlap detection and evaluation

During the annealing process, components are permitted to overlap under the assumption that the future moves will lead the infeasible layouts to superior feasible layouts.

The easiest way to detect an overlap between two components is to detect the overlaps of their projections in every 2-D planes (x-y, y-z, or z-x plane). If there is no overlap between the projections in any 2-D plane, then there is no overlap between these two components. A multi-resolution detection scheme [14, 15] could be adapted to obtain reasonable running time. Rough analyses are performed at the start of the algorithm (low-resolution) and more refine analyses (high-resolution) are performed towards the end of the algorithm where more accurate evaluations are required. The following five figures illustrate the concept of decomposition levels of a component in 2-D plane, and the principle of the resolution levels of an overlap.

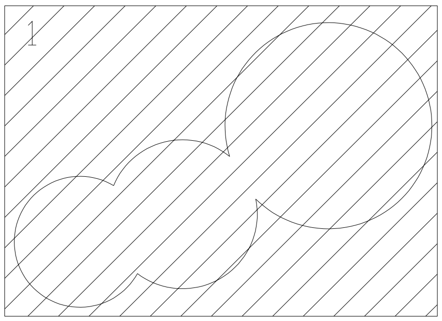

Figure 1 First level of decomposition of 2-D model

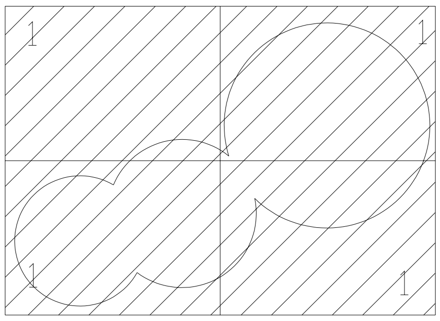

Figure 2 Second level of decomposition of 2-D model

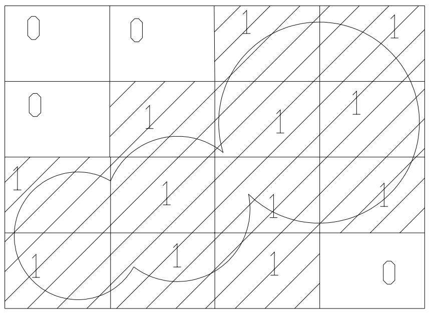

Figure 3 Third level of decomposition of 2-D model

Figures1-3 show three different levels of decomposition for a 2-D model. The first level of decomposition is defined as having a single rectangular, which is used to completely contain the model. Consequently, the higher levels (Figures 2 and 3) are defined as having multi-rectangular which are equally divided from the single rectangular of the first level. The more sub-rectangular are used by the decomposition process, the higher the level is. Further more, it is assumed that each rectangular at each level has its unique color label- white or black (or say binary number 0, 1). The white color indicates that no part of the model sits inside the rectangular; while the black color indicates the entire or partial model rests inside the rectangular.

At each iteration in the optimization process, the annealing algorithm perturbs (*moves*) the position of components and requires overlap detection and evaluation. The amount of overlap is calculated as the sum of the area of overlap projected at each 2-D plane at a specified level of resolution, giving an accurate evaluation at that level of resolution. Figures 4 and 5 show a configuration of an overlap projected at a 2-D plane for two levels of resolution.

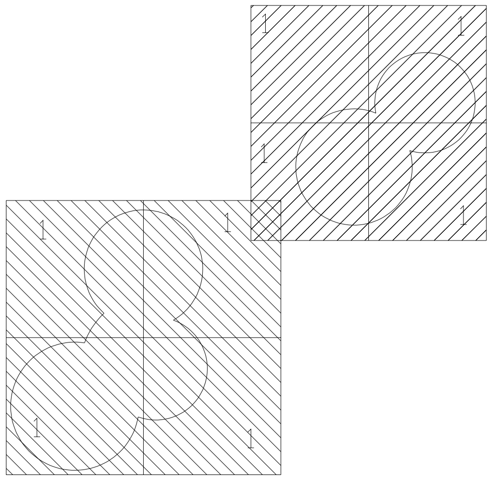

Figure 4 Detected overlap at a low level of resolution

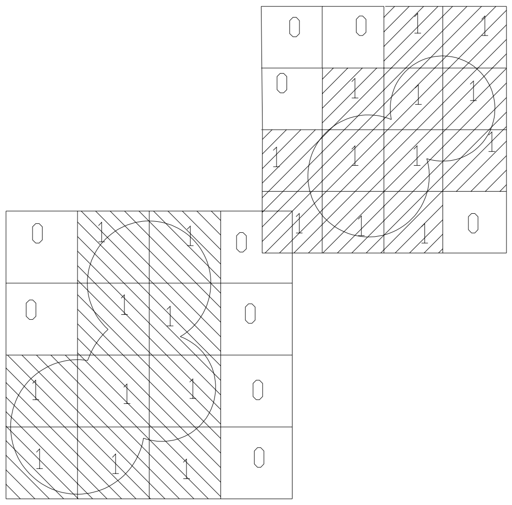

Figure 5 Detected no overlap at a higher level of resolution

Figure 4 represents a lower level of resolution, where two objects are detected overlapping. Figure 5 illustrates the same objects at a higher level of resolution, where no overlap is detected. It is easy to see, that as the resolution level increases, the accuracy of overlap calculation increases as well.

Because of the efficiency and generality, the developed concepts can be used to evaluate the overlaps within the Simulated Annealing process for arbitrary shapes of arbitrary components with any moves.

It is found that early in the annealing process, nearly all layout state perturbations are accepted as new layout states and thus the algorithm does not require a very accurate estimate of the cost function. Conversely, later in the process, layout perturbations are smaller and more accurate evaluations of the cost function are necessary. By changing the resolution level at each move a more or less accurate evaluation of the amount of overlap is determined. At lower levels, less accurate calculations are obtained with minimal computation, while at higher levels, more accurate evaluations take longer to calculate. Thus it allows specify the precision of overlap calculations needed for each stage during the running of the algorithm and therefore significantly save the overall computational time. To determine which resolution level should be used at each stage, the simplest way is to increase the accuracy of the calculations as a function of decreasing of the *temperature*.

4.3 Spatial constraint and annealing violation

There are special relationships between particular components or housing constraints, such as the placement constraints of a component, the assembly constraints, the electronic interference avoidances, the electro-mechanical-interface match-ups, the wiring restrictions, etc. It is necessary to constrain components with respect to the envelope and each other.

For those particular components with restrictions in a given direction (a particular absolute position or orientation with respect to a linear combination of global coordinate axes), it is simple to place the component in a feasible initial position and its *moves* are restricted in such a way that it cannot be moved to an infeasible point or be rotated along an infeasible axis. Similarly, for those components with particular restrictions to the global coordinate axes, the constraints may be violated under the assumption that allows the annealing to 'pass through' infeasible layouts and lead to feasible layouts. The constraints are violated at the beginning of the algorithm and the layouts are pushed into the feasible range as the algorithm processes.

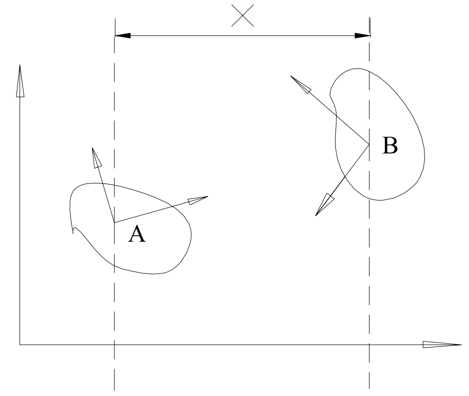

Figure 6 Spatial constraints by centre

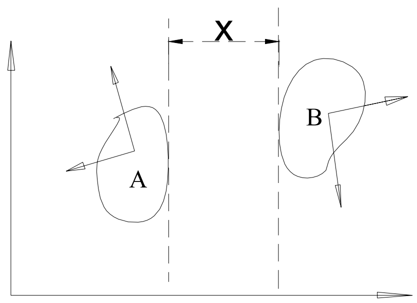

Figure 7 Spatial constraints by extents

In this work, two types of such constraints would be adopted [5] with the employment of violation process, which is constrained by centers or by extents. Constrained by center is to restrict the position of component based upon its origin of coordinate system (Figure 6). For instance, components A and B are restricted with x distance, which is measured from their own origin of coordinate system. Constrained by extent is to restrict the position of all points of the component rather than only its origin (Figure 7). These constraints could be implemented to firstly separate the constraint into x, y, z directions to restrict the parallel transfers of a component and its rotations, then the annealing violation for each component is calculated and a penalty function could be created that penalize the cost function for the constraint violation.

5. Software Implementation and Embedment

The optimization algorithm / procedure would be programmed in Visual C++ [14, 15, 16] and would be embedded into SolidWorks [19, 20, 21] so that the designer can concurrently recall the optimization software while drafting in a CAD environment. The algorithm parameters can be adjusted through the provided User Graphic Interface. All interfaced components can be filled into a '3D-envelope' in a piecewise fashion and produce a minimized 'envelope' representing the design space.

Figure 8 indicates the logical flow chart for an algorithm implementation. Firstly, it creates an initial layout specified by the designer or by using a suitable knowledge-based method. The designer is then asked to input the initial parameters in order to start the algorithm, such as the *initial temperature*, the *final temperature*, the *maximum iterations for each temperature*, the *spatial constraints*, etc. Once the initial settings are specified, the Simulated Annealing process is initiated to calculate overlaps between components and that, in turn, starts the inner *move* loop. If the termination condition -- final temperature -- is reached, then the best solution is taken and the optimization process is stopped. Otherwise the inner *move* loop is continued from the current layout until it reaches the acceptable criteria. The newly accepted layout is further checked at the outer loop to see if the equilibrium condition has been reached. If it is, then the algorithm reduces the temperature following the cooling schedule; starts a new inner *move* loop and repeats the previous steps again. Otherwise, with the same temperature the algorithm returns to the existing inner loop and continues to *move* sets. Obviously, these annealing steps finally lead to the optimal layout.

Below are some major parameters and definitions of the algorithm:

- *Initial temperature* – Our initial temperature T_o is based on the acceptance rate. As we measured overlap in percentage, so the initial temperature T_o is set at 100. If the new layout has 50% worse than the old one at T_o, then the *Probability* $_{\text{accept}}$ =0.606, that means there is still a relatively large chance that the inferior solutions can be accepted at the beginning of the process.
- *Final temperature* -- The final temperature T_f is set at 0.8. If the new solution has 1% worse than the old one at T_f, then the *Probability* $_{\text{accept}}$ =0.287, that means there is not many chance that the inferior solutions can be accepted at the final stage of process.
- *Maximum iterations for each temperature* — To achieve an efficient processing time and capability of finding reasonably good local optimum, we set the maximum number of iterations at 2000.

- *Equilibrium condition* – There is a variety of ways to define if the equilibrium has been achieved. We set an upper boundary for the possible number of neighborhood moves. If there is no improvement in the foregoing 10 moves, we decide that the equilibrium has been attained at that temperature.
- *Cooling schedule* – The performance of this algorithm also depends on the cooling schedule which is essentially the temperature updating function. Two schedules are employed in this work.

  -- Proportional decrement scheme,

  $$T_{k+1} = \alpha T_k, \text{ where } 0<\alpha<1. \qquad (2)$$

  -- Lundy and Mees scheme [20],

  $$T_{k+1} = T_k / (1+\beta T_k), \text{ where } \beta>0, \text{ a suitable chosen parameter.} \qquad (3)$$

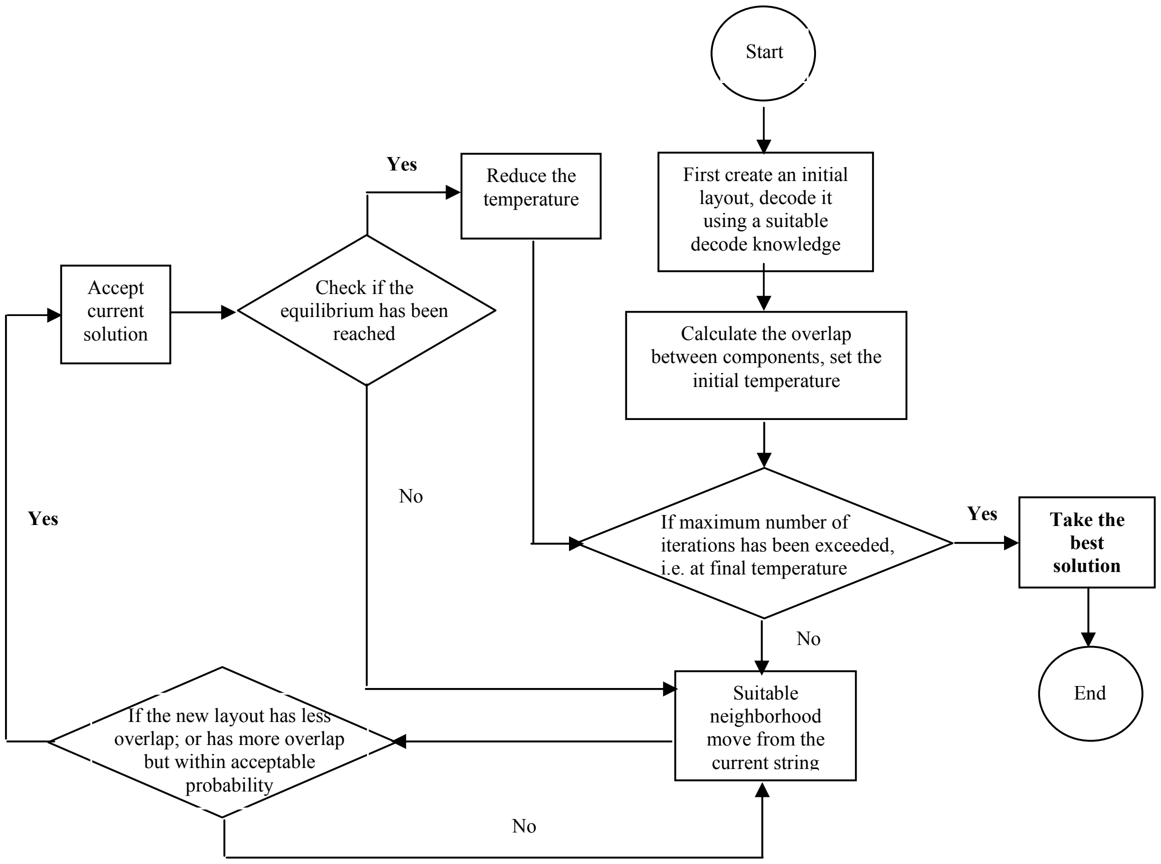
Start
First create an initial layout, decode it using a suitable decode knowledge
Calculate the overlap between components, set the initial temperature
If maximum number of iterations has been exceeded, i.e. at final temperature
Yes
Take the best solution
End
No
Suitable neighborhood move from the current string
If the new layout has less overlap; or has more overlap but within acceptable probability
No
Yes
Accept current solution
Check if the equilibrium has been reached
Yes
Reduce the temperature
No

Figure 8 Logical flow chart of Space Optimization Algorithm / Procedure Using SA

6. Design with Intelligence

Once this optimization algorithm has been embedded in SolidWorks, an intelligent CAD design environment is established ready for design of a wide range of portable mechatronic devices and also other compact equipment, if it requires high density of packing performance. While drafting, the designer could concurrently recall the software, and change the input parameters through the GUI at any time to simultaneously run more than one optimization processes. That intelligent system allows the designer to develop different configurations of assembly and compare its packing performances before selecting a detail design. The development time could be significantly saved and an efficient design procedure could be achieved. More over, with this intelligent system, the designer is enabled to link pre-processed modules or subassemblies into a new envelope to provide integrative tailored modeling solutions for even much complex assembly space in a systematic manner. Figures 9, 10 and 11 are just an illustration of this potential.

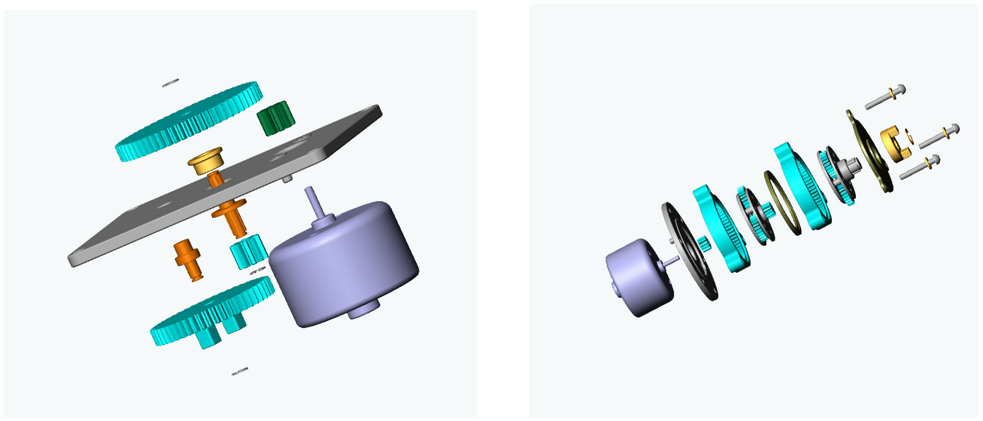

Figure 9 Pre-processed subassemblies of gear train

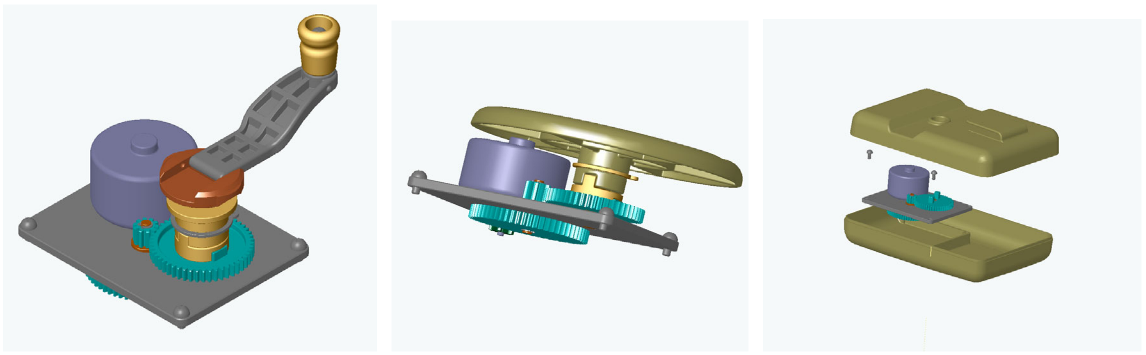

Figure 10 Pre-processed configuration modules

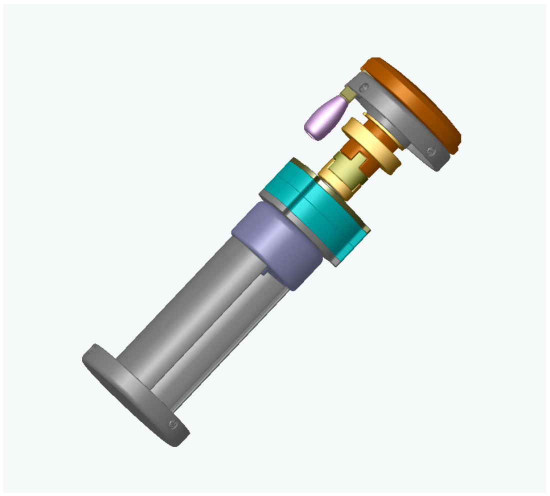
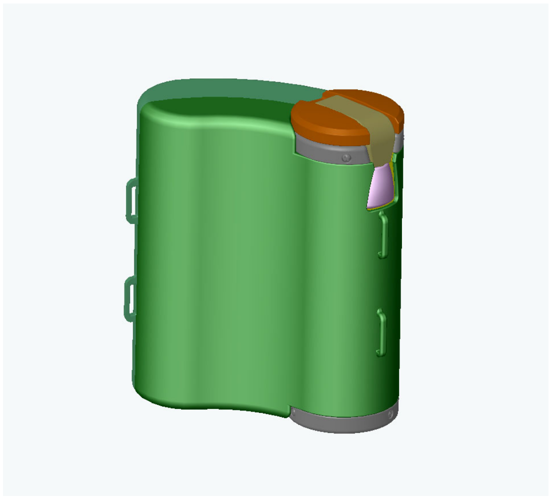

Figure 11 Integrative tailored modeling solutions

7. Conclusions

This paper presented a novel space optimization algorithm for design of mechatronic assemblies based on Simulated Annealing technique. This algorithm can be used as an optimization tool to intelligently minimize the housing space for such devices where a large number of different sections and components are interfaced in the assembly. The optimization algorithm / procedure was in turn programmed in Visual C++ and embedded into SolidWorks environment. All interfaced components can be filled into a '3D-envelope' in a piecewise fashion and intelligently produce a minimized 'envelope' representing the design space.

As an intelligent design tool, SA algorithm / procedure can be used for the optimal design of a wide range of portable devices and also other compact equipment which require high density packing performance. An interesting future work would be to test the algorithm in application to other complex problems and to improve the detection of overlap in order to minimize the computational time.

REFERENCES

1. Emile Aarts, Jan Korst, "Simulated Annealing and Boltzmann Machines: a stochastic approach to combinational optimization and neural computing", John Wiley & Sons Ltd, 1989.
2. P.J.M. van Laarhoven, E.H.L.Aarts, "Simulated Annealing: Theory and Applications", D.Reidel Publishing Company.
3. Kathryn A. Dowsland, "Some Experiments With Simulated Annealing Techniques for Packing Problems", 0377-2217/93, Elsevier Science Publishers B. V.
4. P.Jain, P.Fenyes, et al., "Optimal Blank Nesting Using Simulated Annealing", *Journal of Mechanical Design, 160 / Vol.114, March 1992.*
5. Szykman S, Cagan J. "Constrained three dimensional component layouts using simulated annealing." ASME Journal of Mechanical Design 1997, 119 (1): 28-35.

6. Szykman S, Cagan J. “A simulated annealing approach to three-dimensional component packing.” ASME Journal of Mechanical Design 1995, 117 (2A): 308-14.
7. Szykman S, Cagan J. “Synthesis of optimal non-orthogonal routes.” ASME Journal of Mechanical Design 1996, 118 (3): 419-24.
8. Szykman S, Cagan J, Weisser P. “An integrated approach to optimal three dimensional layouts and routing.” ASME Journal of Mechanical Design 1998, 120 (3): 510-2.
9. Rao RL, Iyengar SS. “Bin-packing by simulated annealing.” Compute Mach Appl 1994; 27(5): 71-89.
10. Han G.C, Na S. J. “two-stage approach for nesting in two-dimensional cutting problem using neural network and simulated annealing.” Journal of Engineering Manufacturing 1996:509-19.
11. Metropolis, N. Rosenbluth, et al. “Equations of state calculation by fast computing machines.” Journal of Chemical Physics, 1953, 21, 1087-1092.
12. Kirkpatrick, S., Gelatt, C.D. Jr. and Vecchi, M.P., “Optimization by simulated annealing”, Science, 1983, 220(4598), 671-679.
13. Sorkin GB. “Efficiennt simulated annealing on fractal energy landscapes.” Algorithmic a 1991; 6:367-418.
14. Meagher, D., “Geometric modeling using octree encoding.” Computer Graphics and Image Processing, 1982. 19(2), 129-147.
15. Aref, W.G. and samet, H., “An algorithm for perspective viewing of objects represented by octrees.” Computer Graphics Forum, 1985, 14(1), 59-66.
16. Steve Holzner, “Fast Track Visual C++ 6.0 Programming”, John Wiley & Sons, Inc. 1998.
17. Pappas Christ, H “Visual C++6: the complete reference” Berkeley, Calif.: Osborne/McGraw-Hill, c1998.
18. Zaratian, Beck, “Microsoft Visual C++ 6.0 programmer's guide.” Redmond, Wash.: Microsoft Press, c1998.
19. Planchard, David C., “Engineering design with SolidWorks 2004 : a step-by-step project based approach utilizing 3D solid modeling”, Schroff Development Corporation, c2004.
20. Lueptow, Richard M. “Graphics concepts with SolidWorks.” Edition: 2nd ed. Date: Upper Saddle River, NJ: Pearson/Prentice Hall,c2004.
21. Tickoo, Sham. “SolidWorks for designers Release 2004.” Date**:** Schererville, IN: CADCIM Technologies, 2004.
22. Lundy, M., Mees, A. “Convergence of an annealing algorithm.” Mathematical Programming, 34, 111-124.
23. Erik K. Antonsson, Jonathan Cagan, “Formal Engineering Design Synthesis”, *Cambridge University Press, 2001.*
24. T.W. Leung, C.H. Yung, Marvin D. Troutt, Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. Computers & Industrial Engineering 40 (2001) 201-214.