Employing Simulated Annealing Algorithms to Automatically Resolve MEP Clashes in Building Information Modeling Models

H.C. Hsu[a] and I.C. Wu[a]

[a]Department of Civil Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan

E-mail: F107141125@nkust.edu.tw, kwu@ nkust.edu.tw

Abstract –

Building Information Modeling (BIM) covers the whole lifecycle of a building and facilitates the coordination of activities in the design, construction, and operation stages. However, during the design stages of pre-construction, it is time-consuming for a BIM project team to resolve design clashes as they integrate models finished by individual team members into a composite master model. To effectively overcome the issue, this study proposes a computer programming system. Based on the application programming interface provided by BIM software, a simulated annealing algorithm is employed to determine the layout modifications to minimize the number of design clashes. In this paper, the mechanical, electrical and plumbing (MEP) systems in a clean room on the first floor of an integrated circuit assembly factory are used to validate the effectiveness of the proposed system. The experimental results reveal the feasibility and effectiveness of the proposed system.

Keywords –

Building information modeling; Clash resolution; Simulated annealing algorithms

1 Introduction

As building information modeling (BIM) software matures, BIM is gradually becoming conventional in both design and construction practice worldwide [1]. As first defined in the National BIM Standard–United States®, a BIM model is a digital representation of the physical and functional characteristics of a facility. As such, BIM serves as a shared knowledge resource for information about a facility, forming a reliable basis for decisions during its life cycle from inception onward [2]. In addition to helping in the design stage, BIM provides decision makers with the ability to make informed decisions across the lifecycle, in the construction stage [3], the project closeout stage [4], and the facility management stage [5]. BIM has also become a platform for project management teams to collaborate [6]. Through the digital portal provided by BIM software, team members such as architects, structural engineers, and mechanical, electrical and plumbing (MEP) engineers can collaborate on a design and share their knowledge of a construction development at the design, construction, and post-construction stages [7]. During the design stages of pre-construction, BIM models finished by team members are integrated into a composite master model, which is then tested to detect design clashes [8], The design clashes defined in [6] are 'positioning errors', where building components overlap each other when original individual designer models are merged. During the construction phase, rework caused by design clashes undetected in the pre-construction stage is usually costly. However, resolving these design clashes is a time-consuming task and is imperative to project performance [9]. Obviously, it presents a great challenge for project team members to ensure there are no clashes in a composite master model within a reasonably short time, even with the use of BIM software.

In addition to the modeling functions, BIM software provides an application programming interface (API) for users to effectively achieve multitudinous applications of BIM models. In previous studies, Mangal and Cheng [10] employed a hybrid genetic algorithm (GA) and the API provided by BIM software to develop an automatic system for the optimization of steel reinforcement in RC buildings. Lin and Lin [4] took advantage of API to propose a final as-built BIM model management system for owners to handle the inspection, modification, and confirmation work beyond project closeout. Based on a repetitive trial-and-error procedure, Xue and Lu [11] presented a novel segmentation-free, derivative-free optimization approach that translates as-built BIMs from two-dimensional images into an optimization problem of fitting BIM components within architectural and topological constraints. Moreover, to evaluate the overall thermal transfer value of the building envelope and the cost of construction, Lim and Majid [12] developed a BIM-GA optimization method by using the functionalities of BIM software, Autodesk Revit, the iterated learning of GA, and the computer programming of PHP. In fact, their method [12] can also be achieved by directly using the API provided by Revit.

To facilitate designer to resolve the clashes, numerous emerging model collaboration systems, such as EXPRESS Data Manager and BIM 360, have been developed to make it possible to have the ability to manage the coordinated workflow required for clash resolution. However, this function still requires human intervention [15].

To automatically resolve the design clashes in a composite master model, this study developed an effectively system by using the API provided by Revit to control building components and adopting a simulated annealing (SA) algorithm [13] to implement iterated learning to simulate coordination cycles. In consideration of the fact that when some clashes are resolved, other clashes may occur, the iterated learning process of the SA algorithm is used. Heuristic optimization methods, such as GA and SA, have been extensively applied to searching the fittest solutions of combinatorial problems. Single thread processing is more efficient for design clashes resolving problem. Therefore this paper adopts SA. The related works such like Hackl, et al. [14] utilized SA to determine the optimal restoration programs for transportation networks. To tackle the search issue in BIM projects. Zeferino, et al. [16] present an efficient simulated annealing (SA) algorithm for solving a regional wastewater system planning model. Focusing on MEP systems, our program detects design clashes with their coordinates and then makes modifications to building components, such as moving or revising them, to gradually minimize the number of design clashes. In the experiment, we tested our system on a real case that occurred during the compilation of a federated BIM model for the MEP systems in the clean room in a factory. The new layout of the MEP systems could be taken as the suggested prototype for the discussion of clash resolution in the design team meeting.

The remainder of this paper is organized as follows: Section 2 introduces the simulated annealing (SA) algorithm. Section 3 explains the proposed system. Section 4 addresses the experimental data and discusses the experimental results. Finally, Section 5 provides the conclusion to this paper.

2 Simulated Annealing Algorithm

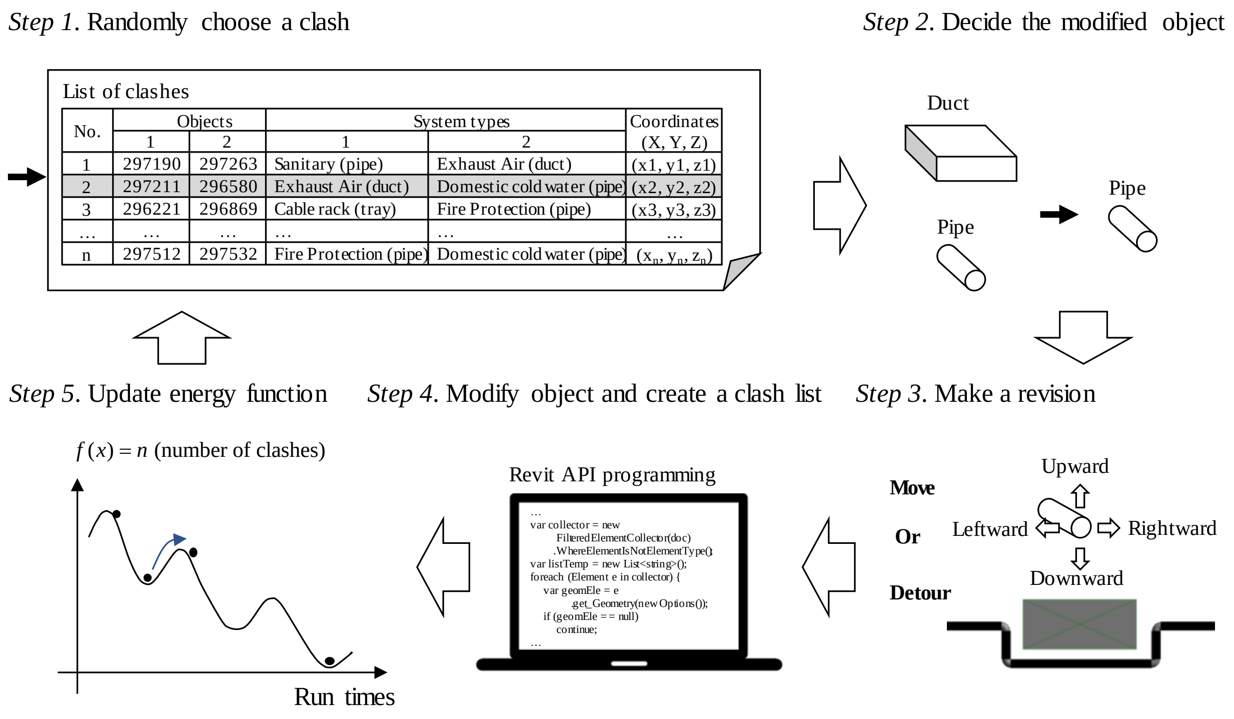
The SA algorithm proposed by Kirkpatrick and Gelatt [13] is an extension of the Monte Carlo (MC) method and is widely applied to approximating the global optimum in combinatorial problems. The name SA comes from annealing in metallurgy, wherein the states of molecular structures of a material are changed by heating and cooling. Heating and cooling the material affects both the temperature and the thermodynamic free energy. SA thus employs MC to generate random samples to simulate the states of a thermodynamic system. Unlike MC, which is a completely random method, SA has a mechanism to control the movement of molecules as the temperature decreases to make a converged learning process. The learning process of SA is briefly described in pseudocode in Figure 1.

```
// $s_0$: a given initial state (solution)
// $iter_{max}$: the maximum iteration number
// $T_0$: a given initial temperature
function SimAnneal($s_0$, $iter_{max}$ , $T_0$)
    $s = s_0$; // Set current states as $s_0$
    $T = T_0$; // Set current temperature $T$ as $T_0$
    $i$=1; // Set the iteration number as 1
    while ($i \le iter_{max}$)
        // Create a feasible neighbor state $s_{new}$
        $s_{new}$ = CreateOneNeighbor($s$);
        // The energy function which updates $T$ according to
        iteration times
        $T$ = UpdateTemperature($i$, $T$);
        // Check if accept $s_{new}$ under $T$
        if Cost($s$) ≤ Cost ($s_{new}$)
            $s = s_{new}$; $i$++;
        else
                // Giving an opportunity of accepting a worse $s_{new}$
            if $P$(Cost($s$), Cost($s_{new}$), $T$) ≥ Random(0, 1)
                $s = s_{new}$; $i$++;
            end if
        end if
    end loop
    return $s$;
end function
```

Figure 1. The pseudocode of simulated annealing

3 The Proposed System Framework

The objective of this study was to develop an effective programming system to automatically resolve the design clashes of MEP systems when BIM models finished by individual team members are integrated into a federated BIM model. The iterated learning process of the proposed system has five steps, as shown in Figure 2. Through the API provided by Revit, attributes of building components such as types, shapes, lengths, widths, and positions (coordinates) can be controllable and revisable, and even clash coordinates can be detected. More information about the use of the Revit API is available on the official Revit website(http://www.revitapidocs.com/). Below, we will first explain the definition of the clash list used in our system, since the clash list is the core of our system. Then the five steps in Figure 2 will be addressed.


Step 1. Randomly choose a clash
Step 2. Decide the modified object
Duct
Pipe
Pipe
Step 5. Update energy function
Step 4. Modify object and create a clash list
Step 3. Make a revision
f (*x*) = *n* (number of clashes)
Run times
Revit API programming
Move
Or
Detour
Upward
Leftward
Rightward
Downward

List of clashes

No.	Objects		System types		Coordinates
	1	2	1	2	(X, Y, Z)
1	297190	297263	Sanitary (pipe)	Exhaust Air (duct)	(x1, y1, z1)
2	297211	296580	Exhaust Air (duct)	Domestic cold water (pipe)	(x2, y2, z2)
3	296221	296869	Cable rack (tray)	Fire Protection (pipe)	(x3, y3, z3)
…	…	…	…	…	…
n	297512	297532	Fire Protection (pipe)	Domestic cold water (pipe)	(x_n, y_n, z_n)

```
…
var collector = new
        FilteredElementCollector(doc)
       .WhereElementIsNotElementType();
var listTemp = new List<string>();
foreach (Element e in collector) {
    var geomEle = e
              .get_Geometry(new Options());
    if (geomEle == null)
        continue;
…
```

Figure 2. The five steps in the iterated learning process of the proposed system

3.1 Definition of Clash List

The core of our system is the clash list, which is created by our programming through the Revit API, and the mission is to minimize the number of detected clashes. An example of a clash list is provided in Table 1, in which there are three main attributes: objects, systems, and three-dimensional coordinates.

Table 1. An example of a clash list used in this study

No.	Objects		System types		Coordinates
	1	2	1	2	(X, Y, Z)
1	297190	297263	Sanitary (pipe)	Exhaust Air (duct)	(x1, y1, z1)
2	297211	296580	Exhaust Air (duct)	Domestic cold water (pipe)	(x2, y2, z2)
3	296221	296869	Cable rack (tray)	Fire Protection (pipe)	(x3, y3, z3)
…	…	…	…	…	…
n	297512	297532	Fire Protection (pipe)	Domestic cold water (pipe)	(x_n, y_n, z_n)

1. The "objects" attribute shows a pair of building components whose geometric shapes intersect. The values of the objects are the identifiers (id) given by Revit.
2. The "systems" attribute lists the MEP systems to which the two intersected objects belong.
3. The "coordinates" attribute is the three-dimensional position of the clash between object 1 and object 2, plotted in three values of X-axis, Y-axis, and Z-axis. Based on a clash coordinate and the shape profiles of the two objects, our program can revise an object or decide on a moving distance of an object, or the whole system, from the original coordinates to a new position to avoid the clash.

3.2 The Operation Principles

The operation principles are used to determine which of the two objects should be modified and what revision options can be selected in a clash instance. There are two principles, the priority of systems and the available revision options. It is notable that these principles can be experience-based, case-based, or country-based.

According to the case examined in this study, the contents of the two principles are briefly described as follows:

1. Based on [13], the priority order of MEP systems and the reasons are listed in Table 2, in which parts of the system names have been revised to be consistent with the words used in Revit. For example,

if the systems of object 1 and object 2 are "Fire Protection (pipe)" and "Domestic cold water (pipe)", respectively, object 2 should be chosen for revision, rather than object 1.

2. The revision options for MEP objects are defined in Table 3, in which the circles denote the action options available for an object. For instance, "Sanitary pipe" has two possible revision actions, "Moving" and "Sloping", while "Duct" only has the "Moving" action. Note that only the revision action "Moving" is available when our system determines to execute a revision on a whole MEP system to which object 1 or object 2 belongs. Moreover, a brief description of the three revision actions are summarized in Table 4.

Table 2. The priority order for sequential comparison process (source [13] and revised in this paper)

System	Priority/special notes
Exhaust Air (duct)	Usually first due to large size of components
Supply Air (duct)	Follows HVAC Dry due to interdependence of these systems
Sanitary (pipe)	Design criteria for slope essential for system performance
Process piping	Takes the first priority if critical to manufacturing process
Fire Protection (pipe)	Most flexible routing, especially small diameter pipe
Domestic hot/cold water (pipe)	Lower priority because less difficult to re-route
Cable rack (tray)	Flexible routing within safety and architectural requirements
Control systems	Flexible routing but must limit bend radius for pneumatic tubes
Telephone/Data communications	Flexible routing but must limit bend radius for fiber optic cables

Table 3. The revision options for MEP objects

Actions	Sanitary pipe	Duct	Cable rack	Other pipes
Moving	○	○	○	○
Revising			○	○
Sloping	○			

Table 4. The descriptions of revision actions

Actions	Descriptions
Moving	Move an object or a system upward, downward, leftward, or rightward to avoid clashes.
Revising	Make a pipe or a cable rack take a roundabout way to avoid clashes.
Sloping	Adjust the slope of a sanitary pipe to avoid clashes. The minimum acceptable ratios of slopes are defined as 1:100 in USA and 2:100 in Japan. This revision action is only applicable to sanitary pipe.

3.3 The Learning Steps

In this subsection, we present the five steps in Figure 2, which are an iterated learning process in our system. Below, the details of the five steps are explained.

Step 1. Randomly select a clash instance from the clash list as a start of the current iterated learning. Suppose that the No. 2 instance in Figure 2 is chosen.

Step 2. For the selected clash instance, decide which of object 1 and object 2 should be chosen for modification according to the priority order of their systems, as mentioned in section 3.2. For the No. 2 instance, for example, since the systems of object 1 and object 2 are "Exhaust Air (duct)" and "Domestic cold water (pipe)", respectively, object 2 should be chosen for revision.

Step 3. Revise the object or its system as decided in *Step 2* through Revit API. Here, we define a parameter $\theta \in [0, 1]$ as the criterion to determine whether the revision is executed on an object or its whole system. Moreover, we also define a random seed (*rs*) drawn from a uniform distribution [0, 1] as a tester. The revision action that will be taken depends on the following conditions:

1. When $rs \le \theta$, the modification will be made on the object. According to Table 3 and the object's system, randomly choose an action to make the modification. For example, the candidate actions for object 2 (a pipe) are "Moving" and "Revising".
2. When $rs > \theta$, the modification will be made on the object's system. Our system will then move the whole system.

After deciding to revise an object or its system, the proposed system will compute the spatial information of the object as a reference for the decision on revision actions. The spatial information is the available distance aggregated in the four directions (up, down, left, and right) of the object. First, our system calculates the minimum moving distances based on the geometric parameters of object 1 and object 2, such as the widths, lengths, and heights, and the clash coordinates obtained from the Revit API. In the example shown in Figure 3, it is better to move the pipe upward or downward rather than leftward or rightward. Then our system will compute the distances between an object and the objects of the other systems in its neighborhood and the limit of vertical clearance. In Figure 4 (a), for example, there is a grey pipe under the black pipe. If the black pipe were moved downward to avoid the original clash, a new clash between the two pipes would occur, as shown in Figure 4 (b). Accordingly, the black pipe should move farther to avoid the new possible clash. The direction having the minimum aggregated distance is adopted for movement, and a construction tolerance (five cm in this study) is added to the aggregated distance to prevent an imprecise construction collision from being caused by insufficient distance. In this study, the action "Revising" is only applicable when the direction is determined to be up or down. Note that once an object is moved, parts of the objects in the same system, such as its branches or the object from which it branches, would need to be accordingly moved and corresponding lengths added or subtracted.

The steps from *Step 1* to *Step 3* are actually the content in the function **CreateOneNeighbor** in SA in Figure 1. The pseudo code of the function **CreateOneNeighbor** is provided in Figure 5.

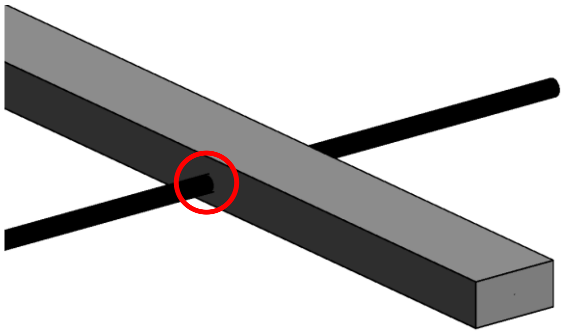


Figure 3. An example of a clash between a duct and a pipe.

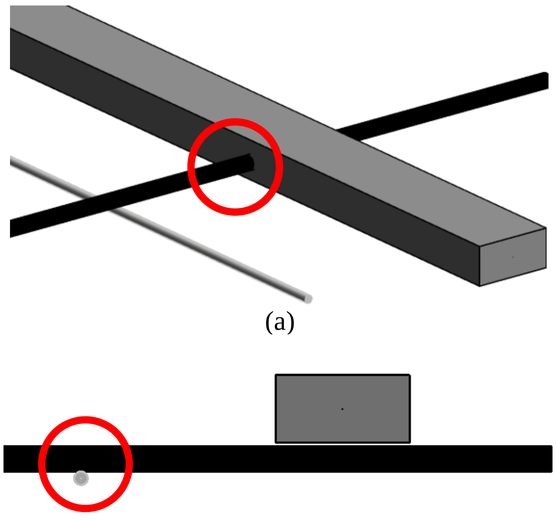


Figure 4. The diagrams of two clashes, (a) an original clash and (b) a new clash caused when the black pipe is moved downward to avoid the original clash

```
// s: a given list of clashes in the form of Table 1
function CreateOneNeighbor(s)
    c; // a randomly selected clash instance from s
    D; // the direction (up, down, left, or right)
    d; // the minimum aggregated distance
    g; // the limit of vertical clearance
    o; // the object or system that needs to be revised in c
    θ; // the criterion to decide to revise an object or a system.
    t; // the construction tolerance
    α; // the criterion to decide moving or revising
    while
        c = GetOneRandomClashInstance(s);
        o = DetermineRevisingObject(c, θ);
        // in some case, d does not exist because of g
        if IsMinDistanceExist(o, t, g, ref D, ref d);
            if D is up or down and o is object
                if Random(0, 1) ≥ α
                    Move(o, D, d);
                else
                    Revision(o, D, d);
                end if
            else
                Move (o, D, d);
            end if
            return false;
        end if
    end loop
end function
```

Figure 5. The pseudocode of creating a neighbor solution for SA

Step 4. According to the revision action decided in Step 3, a revision is executed on an object or its system and then a new list of clashes is output through our program and Revit API. This step is the function **Cost** in SA in Figure 1.

Step 5. Record the number of clashes and update the energy function to decrease temperature.

According to the operation in SA, when temperature is higher, the algorithm has a higher probability to accept a worse solution to escape from the current local area. Nevertheless, as iteration times increase, it becomes less likely that a worse solution will be accepted.

4 The Experiment

In this section, we will briefly introduce the profile of the proposed case and then discuss the experimental results.

4.1 The Proposed Case

The proposed case is the MEP systems in a clean room on the first floor of an integrated circuit (IC) assembly factory, as shown in Figure 6. The clean room has four MEP systems, as listed in Table 5, and the four MEP systems have a total of 50 Revit elements (excluding fittings). When the first MEP design models finished by team engineers were integrated into a federated BIM model, 20 clashes were detected (red circles in Figure 6).

The 20 clashes and their clash coordinates are listed in Table 6. Below, the two places marked 1 and 2 in Figure 6 are presented as examples in Figures 7 and 8, respectively, to show the clash status.

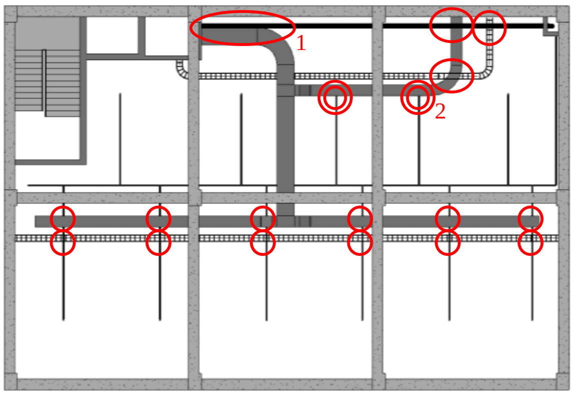


Figure 6. The floor plan of the first floor of the factory

Table 5. The MEP systems in the proposed case

Systems	System types
Supply Air	Mechanical/ Duct
Sanitary	Plumbing/ Pipe
Fire Protection (water)	Plumbing/ Pipe
Electrical	Electrical/ Cable Tray

Table 6. The clashes and their coordinates in the proposed case

No.	Objects		System types		Clash coordinates (X, Y, Z)
	1	2	1	2	
1	367903	371380	Sanitary (pipe)	Cable rack (tray)	(41.819132952, 22.539205146, 10.170603675) (41.819132952, 23.100470191, 10.170603675)
2	370451	367722	Fire Protection (pipe)	Cable rack (tray)	(-22.859319532, -8.881478564, 10.006561680) (-22.661402865, -8.881478564, 10.006561680)
3	370473	367722	Fire Protection (pipe)	Cable rack (tray)	(-8.216683959, -8.881478564, 10.006561680) (-8.018767292, -8.881478564, 10.006561680)
4	370618	367722	Fire Protection (pipe)	Cable rack (tray)	(7.923771212, -8.881478564, 10.006561680) (8.121687878, -8.881478564, 10.006561680)
5	370639	367722	Fire Protection (pipe)	Cable rack (tray)	(22.447435371, -8.881478564, 10.006561680) (22.645352037, -8.881478564, 10.006561680)
6	370660	367722	Fire Protection (pipe)	Cable rack (tray)	(35.638009583, -8.881478564, 10.006561680) (35.835926250, -8.881478564, 10.006561680)
7	370681	367722	Fire Protection (pipe)	Cable rack (tray)	(48.197120136, -8.881478564, 10.006561680) (48.395036803, -8.881478564, 10.006561680)
8	367101	367903	Supply Air (duct)	Sanitary (pipe)	(-2.216012647, 22.460462668, 9.514435696) (6.595010975, 22.460462668, 9.690656168)
9	367107	370627	Supply Air (duct)	Fire Protection (pipe)	(9.595010975, -1.361906094, 10.006561680) (12.219682891, -1.361906094, 10.006561680)
10	367130	370639	Supply Air (duct)	Fire Protection (pipe)	(22.546393704, -7.677522892, 10.006561680) (22.546393704, -6.037102945, 10.006561680)
11	367130	370660	Supply Air (duct)	Fire Protection (pipe)	(35.736967916, -7.677522892, 10.006561680) (35.736967916, -6.037102945, 10.006561680)
12	367130	370681	Supply Air (duct)	Fire Protection (pipe)	(48.296078469, -7.677522892, 10.006561680) (48.296078469, -6.037102945, 10.006561680)
13	367191	370451	Supply Air (duct)	Fire Protection (pipe)	(-22.760361198, -7.677522892, 10.006561680) (-22.760361198, -6.037102945, 10.006561680)
14	367191	370473	Supply Air (duct)	Fire Protection (pipe)	(-8.117725625, -7.677522892, 10.006561680) (-8.117725625, -6.037102945, 10.006561680)
15	369883	370599	Supply Air (duct)	Fire Protection (pipe)	(18.605084296, 12.252387737, 10.006561680)
16	369883	370700	Supply Air (duct)	Fire Protection (pipe)	(31.106234416, 12.252387737, 10.006561680)
17	369883	371780	Supply Air (duct)	Fire Protection (pipe)	(31.106234416, 12.516579798, 9.820465021)
18	369883	371812	Supply Air (duct)	Fire Protection (pipe)	(18.605084296, 12.516579798, 9.820465021)
19	369891	367695	Supply Air (duct)	Cable rack (tray)	(9.595010975, 15.232694664, 9.514435696) (12.219682891, 15.232694664, 9.514435696)
20	369908	367903	Supply Air (duct)	Sanitary (pipe)	(36.007371265, 22.819837668, 10.278903374) (37.648508636, 22.819837668, 10.311726121)

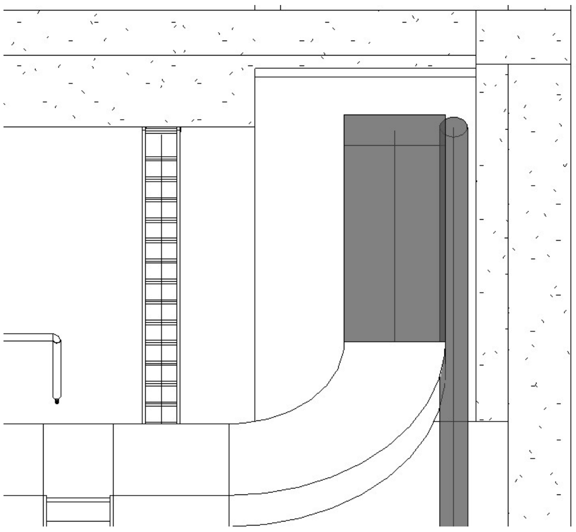


Figure 7. The clash between a duct and a sanitary pipe

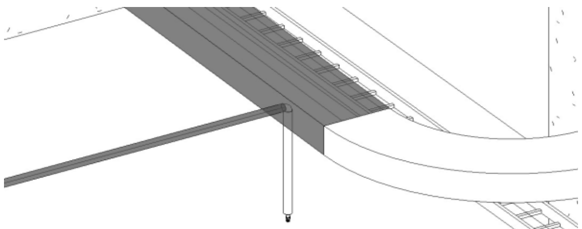


Figure 8. Two clashes between a duct and two pipes

4.2 The Experimental Results

The objective of the proposed system was to resolve the MEP clashes listed in Table 6. The strategy of this study was to set moving as the first priority, i.e., α in Figure 5 is 0.95. The experimental results are summarized in Table 7 and drawn in Figure 7.

Table 7. The details of experimental results

Runs	Clash Numbers	Time (Seconds)
1	14	4.410
2	10	5.822
3	8	5.757
4	2	5.916
5	1	5.788
6	0	5.746
		33.440

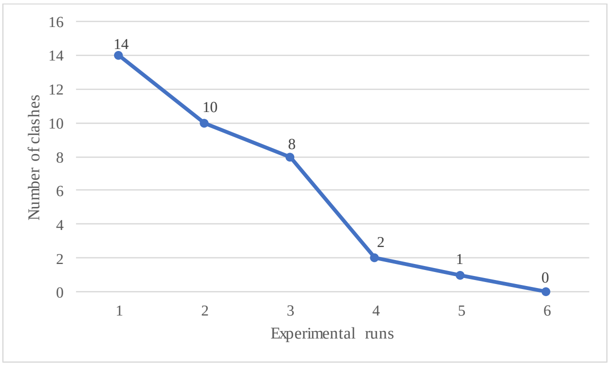


Figure 9. The trend of experimental results

As shown in Figure 9, the clashes were resolved very quickly, in 33.44 seconds. In the first run, six of the twenty clashes were resolved. Although the proposed case is rather simple, it indicates that the proposed system can provide BIM team members an initial reference for further discussions on clash resolution. For comparison with the two examples in Figures 7 and 8, the automatic resolutions are shown in Figures 10 and 11.

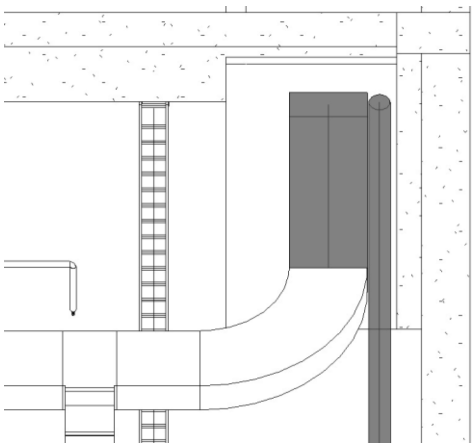


Figure 10. Automatic resolution of a clash between a duct and a sanitary pipe

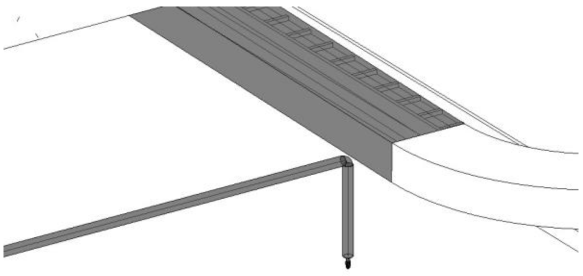


Figure 11. Automatic resolution of two clashes between a duct and two pipes

5 Conclusion

During the design stages of pre-construction, it is time-consuming for a BIM project team to resolve design clashes as they integrate models finished by individual team members into a composite master model. To automatically resolve design clashes, we designed a programming system by employing SA and API provided by Revit. In this paper, a real case of the MEP systems in a clean room in an IC-Assembly factory was employed. In the case, twenty design clashes were detected when the BIM models were merged. The case was used to evaluate the effectiveness and feasibility of the proposed system. The experimental results showed that the twenty design clashes were automatically resolved within a very short time. The revised BIM model can serve as a reference for team members in discussions of how to resolve design clashes. Although the experimental results indicated that design clashes can be automatically resolved in this way, the proposed system still needs more specific guidelines to ensure suitable revisions of building components, rather than random revisions.

References

[1] Liu, Y., van Nederveen, S. and Hertogh, M. Understanding effects of BIM on collaborative design and construction: An empirical study in China. *International Journal of Project Management, 35*(4), 686-698, 2017.

[2] States), N. N. B. S.-U. An Authoritative Source of Innovative Solutions for the Built Environment, 2015.

[3] Love, P. E. D., Liu, J., Matthews, J., Sing, C.-P. and Smith, J. Future proofing PPPs: Life-cycle performance measurement and Building Information Modeling. *Automation in Construction, 56*, 26-35, 2015.

[4] Lin, Y.-C., Lin, C.-P., Hu, H.-T. and Su, Y.-C. Developing final as-built BIM model management system for owners during project closeout: A case study. *Advanced Engineering Informatics, 36*, 178-193, 2018.

[5] Wetzel, E. M. and Thabet, W. Y. Utilizing Six Sigma to develop standard attributes for a Safety for Facilities Management (SFFM) framework. *Safety Science, 89*, 355-368, 2016.

[6] Pärn, E. A., Edwards, D. J. and Sing, M. C. P. Origins and probabilities of MEP and structural design clashes within a federated BIM model. *Automation in Construction, 85*, 209-219, 2018.

[7] Ciribini, A. L. C., Mastrolembo Ventura, S. and Paneroni, M. Implementation of an interoperable process to optimise design and construction phases of a residential building: A BIM Pilot Project. *Automation in Construction, 71*, 62-73, 2016.

[8] Bhagwat, P. and Shinde, R. Clash Detection: A New Tool in Project Management. *International Journal of Scientific Research in Science, Engineering and Technology, 2*(4), 193–197, 2016.

[9] Lee, G. and Kim, J. W. Parallel vs. Sequential Cascading MEP Coordination Strategies: A Pharmaceutical Building Case Study. *Automation in Construction, 43*, 170-179, 2014.

[10] Mangal, M. and Cheng, J. C. P. Automated optimization of steel reinforcement in RC building frames using building information modeling and hybrid genetic algorithm. *Automation in Construction, 90*, 39-57, 2018.

[11] Xue, F., Lu, W. and Chen, K. Automatic Generation of Semantically Rich As-Built Building Information Models Using 2D Images: A Derivative-Free Optimization Approach. *Computer-Aided Civil and Infrastructure Engineering, 33*(11), 926-942, 2018.

[12] Lim, Y. W., Majid, H. A., Samah, A. A., Ahmad, M. H., Ossen, D. R., Harun, M. F. and Shahsavari, F. BIM and Genetic Algorithm Optimisation for Sustainable Building Envelope Design. *International Journal of Sustainable Development and Planning, 13*(1), 151-159, 2018.

[13] Korman, T. M. and Tatum, C. B. *Development of a Knowledge-Based System to Improve Mechanical , Electrical , and Plumbing Coordination.* Paper presented at the Technical Report No. 129, Centre for Integrated Facility Engineering, Stanford University, CA, 2001.

[14] J. Hackl, B.T. Adey, N. Lethanh, Determination of Near-Optimal Restoration Programs for Transportation Networks Following Natural Hazard Events Using Simulated Annealing, Computer-Aided Civil and Infrastructure Engineering 33 (8) (2018) 618-637.

[15] M.T. Shafiq, J. Matthews, S.J.J.o.I.T.i.C. Lockley, A study of BIM collaboration requirements and available features in existing model collaboration systems, 18 (2013) 148-161.

[16] J.A. Zeferino, A.P. Antunes, M.C. Cunha, An Efficient Simulated Annealing Algorithm for Regional Wastewater System Planning, 24 (5) (2009) 359-370.