Fall 2019

Developing a Computational Framework for a Construction Scheduling Decision Support Web Based Expert System

Feroz Ahmed
University of Southern Mississippi

DEVELOPING A COMPUTATIONAL FRAMEWORK FOR A CONSTRUCTION SCHEDULING DECISION SUPPORT WEB BASED EXPERT SYSTEM

by

Feroz Ahmed

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Computing Sciences and Computer Engineering
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. Bikramjit Banerjee, Committee Chair
Dr. Dia Ali
Dr. Tulio Sulbaran
Dr. Beddhu Murali
Dr. Ras Pandey

Dr. Bikramjit Banerjee Committee Chair	Dr. Andrew H. Sung Director of School	Dr. Karen S. Coats Dean of the Graduate School

December 2019

Published by the Graduate School

ABSTRACT

Decision-making is one of the basic cognitive processes of human behaviors by which a preferred option or a course of action is chosen from among a set of alternatives based on certain criteria. Decision-making is the thought process of selecting a logical choice from the available options. When trying to make a good decision, all the positives and negatives of each option should be evaluated. This decision-making process is particularly challenging during the preparation of a construction schedule, where it is difficult for a human to analyze all possible outcomes of each and every situation because, construction of a project is performed in a real time environment with real time events which are subject to change at any time. The development of a construction schedule requires knowledge of the construction process that takes place to complete a project. Most of this knowledge is acquired through years of work/practical experiences. Currently, working professionals and/or students develop construction schedules without the assistance of a decision support system (that provides work/practical experiences captured in previous jobs or by other people). Therefore, a scheduling decision support expert system will help in decision-making by expediting and automating the situation analysis to discover the best possible solution. However, the algorithm/framework needed to develop such a decision support expert system does not exist so far.

Thus, the focus of my research is to develop a computational framework for a web-based expert system that helps the decision-making process during the preparation of a construction schedule. My research to develop a new computational framework for construction scheduling follows an action research methodology. The main foundation components for my research are scheduling techniques (such as: Job Shop Problem), path-

finding techniques (such as: travelling salesman problem), and rule-based languages (such as JESS). My computational framework is developed by combining these theories. The main contribution of my dissertation to computational science is the new scheduling framework, which consists of a combination of scheduling algorithms that is tested with construction scenarios. This framework could be useful in more areas where automatic job and/or task scheduling is necessary.

Keywords: construction scheduling, algorithms, expert system, automation.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Bikramjit Banerjee for his continuous support of my PhD study and related research. He has been a tremendous mentor, always helping me in challenging times and steering me in the right direction. I would like to express my sincere and whole-hearted appreciation to Dr. Dia Ali for encouraging my research and helping me through all the tough situations. His advice on both research as well as on my career have been invaluable. I will forever be grateful to him. This research would not have been possible without the insights, assistance, encouragement, and support of Dr. Tulio Sulbaran. He is a remarkable guide and mentor with valuable ideas. I would also sincerely like to thank rest of my committee: Dr. Beddhu Murali and Dr. Ras Pandey for their insightful comments and encouragement, but also for their hard questions, which helped me to widen my research from various perspectives. I would also like to thank Dr. Wonryull Koh for the supervision and guidance in my research.

My sincere thanks also go to the School of Construction, who provided me access to the resources and equipment required for my work. Without their support, it would not have been possible to work on this research effortlessly.

I am grateful to all my family and friends, who have provided me moral and emotional support along the way.

DEDICATION

I dedicate this dissertation to my grandmother Chotijaan, who taught me determination, my father Ahmed Basha, who taught me dedication, my mom F.N.Sultana, who taught me patience and love, and my brother Sharief, who taught me to be courageous.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

BIM	Building Information Modelling
CPM	Critical Path Method
IP	Integer Programming
ILP	Interval linear programming
JSP	Job Shop Problem
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
TSP	Travelling Salesman Problem
REILP	Risk Explicit Interval linear programming
ICDMA	Interactive Construction Decision Making Aid
PERT	Program Evaluation and Review Technique

CHAPTER I - INTRODUCTION

In the field of construction, project schedule is a vital part that helps project managers to efficiently plan the development of a project. Project schedule not only includes the timeframe for each activity, but also the management of resources like costs, labor, machines etc. Scheduling consists of integrating a plan within the time-period and assigning time durations and resources to each task. A schedule lists the sequence of tasks to be completed. Project managers use the schedule to determine when and how many workers, materials, and equipment are needed. The research for my dissertation on developing a computational framework for a construction scheduling decision support web based expert system is grounded mainly on these following four areas (figure 1):

1. Project/Construction Scheduling
2. Jess Expert System
3. Scheduling Algorithms
4. Building Information Modeling

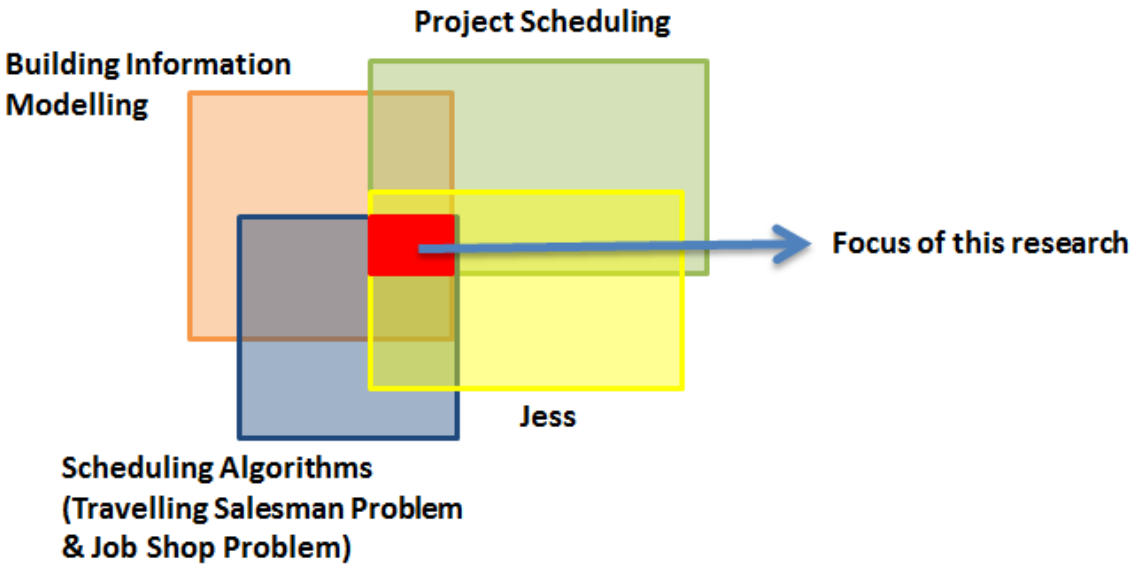

Figure 1 Frameworks and focus of research

1.1 Project Scheduling

Planning and scheduling are very important parts of a project in construction management. Scheduling of a project in construction is to make the final work plan fit to the time scale. In particular, it describes the durations and the order of the construction activities that are temporally consistent. A construction project schedule is one of the most important tools for project managers in the Architecture, Engineering, and Construction (AEC) industry that helps them to track and manage the time, cost, and quality of the projects. Developing project schedules is difficult, since it is heavily dependent on project planner's knowledge of work packages, on-the-job-experience, planning capability and oversight [5]. Scheduling of a project is a very critical tool to monitor the project daily, and to keep a track of the project's progress [10]. Project/construction schedules should be executed based on the criteria that are most important for construction project effectiveness (especially time-cost relation). Irrespective of how construction project is attained, duration and cost are two of the many key factors determining project's economic efficiency and fulfillment of the owner's needs and requirements [8]. Maintaining a construction project schedule is one of the most important parts of a project. A properly executed construction schedule can help manage materials, labor and equipment. A schedule lists the sequence in which tasks should be completed. Project managers and contractors use the construction schedule to determine the availability of the resources and manage them accordingly. Most of the schedulers make a construction schedule based on their experience gained from previous projects. Construction schedules are required to determine the quantity of resources required for a project [27]. Construction engineers and managers are usually more inclined to focus on the project cost, but not duration, especially when reduction of

the project duration requires additional usage of resources. Resources are the most influential constraints in construction as they determine the feasibility of a project schedule and help evaluate if it is optimal. The selection of resources (labor and equipment) is the most important part of scheduling and should be considered in compliance with site restrictions and the work to be undertaken [8]. As projects are unique in nature, the creation of a schedule for construction tasks by a planner, for example, should consider an array of conditions such as technological and organizational methods and constraints, as well as the availability of resource to ensure that a client's needs and requirements in terms of time, cost and quality are met [8]. Construction scheduling is mainly about optimally sequencing activities over time and allocating resources accordingly. Allocating more resources reduces the project duration but may increase the project cost at the same time. Activity dependency, time, cost and resources are the constraints normally considered when scheduling under the auspices of traditional project management. Activity dependency or precedence relationship is the most basic constraint that exists in construction projects. In a construction process, an activity cannot start until all its precedence activities are completed. Irrespective of how construction project is attained, duration and cost are the two of the many key factors determining project's economic efficiency and fulfillment of the owner's needs and requirements [8]. The construction schedule also communicates means and methods, as well as planned sequences and timing for a project. The key processes that are common in all scheduling techniques include planning, controlling, and managing. The scheduling process provides the contractor with a more thorough and structured planning process while they review the plans and determine the sequence for building the project [44]. A proper scheduling system also helps us achieve high efficiency

and complete the project in a reasonable timeframe and within the budget. Main part of project management is properly managing resources to complete tasks efficiently, on schedule, and on budget [40]. Scheduling resources that include people, equipment, or facilities is often a complex task. Availability of resources can help in making flexible schedules. Based on the resources, we can have an estimate of cost and deadlines [40]. Lack of resources/inadequate resources may result in resource overloading. Realistic construction activities/schedule cannot be developed without considering resources [14].

1.2 Scheduling Algorithms

We now discuss some of the classic computational problems in scheduling/planning, viz., Traveling Salesman Problem and Job Shop Problem. We leverage these problems to model various components of the construction-scheduling problem.

1.2.1 Travelling Salesman Problem (TSP)

It is a mathematical problem that involves a salesman who must make a tour to a number of cities using the shortest path available and visit each city exactly once and return to the original starting city. It is a classic algorithmic problem in the field of computer science that focuses on optimization. It is also considered as one of the most intensively studied problems in computational mathematics [46]. TSP optimization algorithm has several applications such as planning, logistics, the manufacture of microchips and DNA sequencing, vehicle routing, order picking from warehouses [33].

Although, some research was done in applications of TSP in Construction management such as the one shown in Table 1, to the knowledge of the researchers no in-

depth experimental research has been done in implementing TSP for construction scheduling.

Table 1 Sample Applications of TSP in Construction Management

Application	**Description**
Handling Operations with Tower Cranes	Optimal scheduling of crane operations not only has direct cost savings, but also results in indirect cost saving by minimizing the idle time of equipment and crew on the job site as well as the downstream delays in the job process. [47]
Construction of roads	TSP solution can be of tremendous help while laying out a path to start the construction of roads connecting multiple cities. It provides the most efficient route to follow the construction and cover each city with minimal wastage of time and resources. [28].

1.2.2 The Job Shop Problem (JSP)

Job Shop Problem is a common scheduling problem technique, in which multiple jobs are processed on several machines. Each job consists of a sequence of tasks, which must be performed in a given order and each task must be processed on a specific machine. This scheduling technique comes with the following constraints. [40]

1. No task for a job can be started until the previous task for that job is completed.
2. A machine can only work on one task at a time.
3. A task, once started, must run to completion.

There are two types of constraints for the job shop problem [41]:

1) **Conjunctive constraints** — these arise from the condition that for any two consecutive tasks in the same job, the first task must be completed before the second task can be started.
2) **Disjunctive constraints** — these arise from the restriction that a machine cannot work on two tasks at the same time.

1.3 Expert Systems

The concept of expert systems was initially developed by Professor Edward Feigenbaum in the 1970s. Development of expert systems has been revolutionary in many industries like financial services, telecommunications, healthcare, customer service, transportation, video games, manufacturing, and aviation [18]. Usually, an expert system integrates a knowledge base containing accrued experience with a rule engine. A rule engine is a computer program that performs a certain task based on a certain situation described in the program. Expert systems have been revolutionizing the research and development of every field of research for several decades. Expert system is one of the extended research domains of artificial intelligence. An expert system is a software program that uses artificial intelligence techniques and databases of expert knowledge to offer advice or make decisions. An expert system applies a set of rules to each particular situation mentioned in the program and gives the output accordingly [37]. The expert system systematically reviews its knowledge to find the specific conditions that will satisfy the problem statement. This ability to review a collection of information and draw conclusions about stated problems is what makes the expert system different from traditional data intensive computer programs [14]. Expert systems generate solutions for

complex and difficult problems by reasoning about knowledge. These systems are designed mainly using IF-THEN structures instead of regular practical codes [35]. The initial development of the expert systems occurred in the 1970s and then became more mature in the 1980s [16]. Expert systems incorporate decision-making processes and can be considered as the mechanization of human thinking. Expert Systems use various methodologies that fall under the following categories: rule-based systems, knowledge-based systems, neural networks, fuzzy expert systems, object-oriented methodology, case-based reasoning, system architecture, intelligent agent systems, database methodology, modeling, and ontology [31]. Expert systems use domain specific knowledge and heuristics to perform many of the functions of a human expert. The components of expert systems are: 1- The knowledge base; 2- the short-term memory; 3- the inference engine; 4- the explanation module; and 5- the knowledge acquisition module [9, 41]. The following is a brief description of each of the five expert system components adopted nearly verbatim from [9, 41]:

1. The knowledge base: contains general information as well as heuristic knowledge. For rule-based systems, this knowledge is represented in the form of IF (condition) THEN (action) rules. Rules may be in the form of situation/action, premise/conclusion or antecedent/consequent relationships [9, 41].
2. The short-term memory: is a dynamic database that stores the temporary facts [9, 41].

3. The inference engine (or executor): is responsible for the execution of the system through manipulation of the rule base and the short-term memory [9, 41].
4. Explanation module: is not a separate entity [9, 41]. Every expert system needs to explain reasoning behind its conclusions like:
    i. How it arrived at a specific decision?
    ii. Why an alternative decision was not reached?
    iii. How a piece of information was used or why the information was ignored?
    iv. What decisions were made for the various sub-problems? and
    v. What are the current actions of the system?
5. The knowledge acquisition module: As the system is demonstrated and put into practice, experts will contribute additional rules and suggestions to augment the knowledge base. An expert system should be capable of adding rules to the knowledge base in a simple and graceful fashion. [9, 41].

Expert systems may include having the power of a database management system that is capable of efficient storage, processing, and retrieval of data. They may also include the ability of a decision support system, which allows for efficient data analysis and trend forecasting. By the use of collection of information, the expert systems will use knowledge, experience and rules of thumb to solve the problems [7]. Expert systems provide high quality experience, domain specific information, apply heuristics, and perform forward or backward reasoning, accommodating uncertainty and explanation capability. For information representation techniques, forward and backward chaining rules are used. An

expert system is developed to imitate the human expert's decision-making ability in a particular domain such as construction management or any other field of knowledge, where there is a shortage of engineers that can comprehend or experts that can give advices and explanations [16]

1.3.1 JESS Expert System:

JESS is a rule engine and scripting environment written entirely in Oracle's Java language [21]. JESS uses an enhanced version of the Rete algorithm to process rules [6]. Rete algorithm is a very efficient mechanism for solving the difficult many-to-many matching problem. A normal JESS program includes facts, rules, functions and template (as shown in Figure 2). The following are the definitions/description of each of them:

- Rules: A JESS rule is something like an if... then statement in a procedural language.
- Facts: a collection of knowledge nuggets used by the rules
- Template: A template describes the characteristics of the facts. Every fact has a template. A template has a name and a set of slots.
- Functions: are a combination of operations and comments executed together. Functions are defined in the JESS rule language using the 'deffunction' construct.

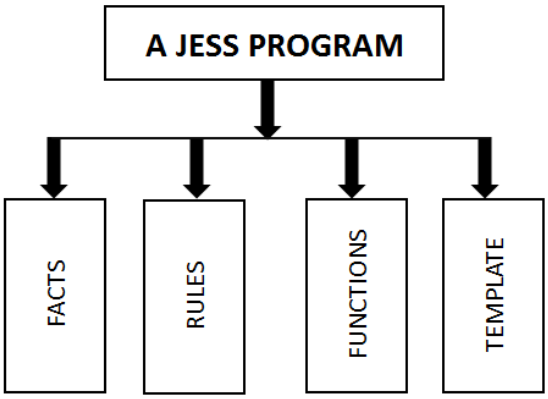

Figure 2 Structure of JESS Expert System program

1.4 BIM

Building Information Modeling (BIM) is a 3D model-based process that equips architecture, engineering, and construction professionals with the insight and tools to more efficiently plan, design, construct, and manage buildings and infrastructure. BIM is a digital representation of physical and functional characteristics of a facility/project. It is a shared knowledge resource for information about a facility/project forming a reliable basis for decisions during its life cycle [11]. BIM data can also be seen as files, which can be extracted, exchanged or networked to support decision-making regarding a building or other built asset. BIM models not only contain architectural data, but the full depth of the building information including data related to the different engineering disciplines such as the load-bearing structures, all the ducts and pipes of the different building systems and even sustainability information [22].

A construction project includes the physical activities requiring resources (such as labor, materials and equipment), but more importantly it also includes the entire scope of activities from conception to realization to successful execution of the project. In today's intensely time driven business environment, superior planning, scheduling, and controlling are very important. The development of a construction schedule requires comprehensive knowledge of the construction process that takes place to complete a project. Most of this knowledge is acquired through years of work/practical experiences. The completion of each project activity requires resources including equipment, materials and labor that can be included in the construction schedule. A well-structured construction schedule minimizes the idle time of available resources and ensures that all parts of the project are completed on time and within the budget. Construction scheduling involves key aspects

such as organizing and controlling the resources, coordinating with different sub-contractors, and directing the resources in the correct manner to finish the project. Successful execution of the project involves the effective planning, scheduling and management of resources. Planning and scheduling of activities and resources is one of the most important tasks of a construction project. A properly prepared construction schedule provides an opportunity to efficiently manage the project and to convey to stakeholders how the construction company is planning to meet its obligations. However, preparing good schedules is a time-consuming process. New innovative technologies might be the key to develop a decision support system that could reduce the time needed to prepare a good schedule. Decision support system is an application or a program that evaluates the data and presents it to the user in a way that could be useful in the decision-making process. Most of the decision support systems include an expert system, which is a computer software that uses artificial intelligence to match the decision-making ability/thinking of a human expert.

CHAPTER II - BACKGROUND

An Expert System can be defined as an interactive and reliable computer-based decision-making system which uses both facts and heuristics to solve complex decision-making problems. Expert systems may include having the power of a database management system that is capable of efficient storage, processing, and retrieval of data. They may also include the ability of a decision support system, which allows for efficient data analysis and trend forecasting. By the use of collection of information, the expert systems will use knowledge, experience and rules of thumb to solve the problems [24]. Expert system mainly relies on the knowledge base of an expert and the inference/rule engine. Telecommunication, financial services, healthcare, transportation, manufacturing and aviation are some of the fields where expert systems are in extensive use [48]. Knowledge-based expert systems are programs that can undertake intelligent tasks that are usually performed by highly skilled people. Expert systems use domain specific knowledge and heuristics to perform many of the functions of a human expert. In construction, the expert system allows the user to model the time-cost relationship of the tasks.

Expert systems in constructing industry have been evolving since 1980s. There are numerous theories on decision making in construction projects and several expert systems are deployed in construction industry. Below are some of the expert systems in the construction industry that are active [4].

- COMIX is a rule and frame based expert system, which provides suggestions on the design of normal weight concrete mixes. This system calculates the amount of cement by suggesting water/cement ratio from a

specified strength. Finally, the volume of coarse aggregate and sand is calculated and displayed by the system [2].

- BETVAL is an expert system whose main function is to help the construction site staff for selecting the type of fresh concrete ordered from the ready-mix concrete plant. This system is mainly based on the knowledge of appropriate concreting techniques, compressive strength class, selecting maximum size aggregate and concrete consistency based on type of structure [31].
- BIDEX is a rule based expert system, which gives suggestions on the design of bid decision. Construction contractors use this expert system for making bid decision [8].
- AMADEUS rule-based expert system is designed for assisting building inspectors during emergency post-earthquake damage assessment. The system records field inspection data and makes recommendations concerning the safety of buildings that have been subjected to earthquake damage [1].

These are just a few of the known expert systems that are being actively used in the field of construction. However, none of these expert systems are designed to develop a schedule automatically for a construction project.

2.1 Literature Review on Decision making Theories

Decision making process in construction projects is a struggling debate for project managers because the decisions must be made based on the criteria that often conflict; decisions need to be adjusted to the cultural, organizational, and social environments

surrounding projects, as these factors have direct impact on the development of the project [46]. There are several theories and methods of decision making that were introduced to the field of construction. Some of the known theories are listed below:

Game theory: As described by Nash, Game theory is the study of strategic decision-making. Game theory is the study of mathematical models that can be applied to describe the decision-making processes in innovation projects in which multiple decision makers are involved. Game theory distinguishes various types of theoretical models to describe decision situations that differ in the sequence of actions, amount of information and type of pay-off: strategic games, extensive games with perfect information, extensive games with imperfect information and coalitional games. [14, 25]

Causal Decision Theory: Lewis described Causal Decision Theory as a theory that endorses principles of rational choice that attend to an act's consequences and that it is a branch of decision theory which advises an agent to take actions that maximizes the causal consequences on the probability of desired outcomes. Causal decision theory advises decision makers to make the decision with the best expected causal consequences. Contrary to Evidential Decision Theory, Causal Decision Theory focuses on the causal relations between one's actions and its outcomes, instead of focusing on which actions provide evidences for desired outcomes. [35]

Evidential decision theory: Egan stated that evidential decision is a branch of decision theory [21], which advises an agent to take actions, which are, conditional on it happening, maximizes the chances of the desired outcome. ED theory, contrary to causal decision theory, believes the best option conditional on having

chosen it is the one with the best outcome. Evidential decision theory says that the action that it's rational to perform (ignoring the possibility of ties) is the one with the greatest expected utility – the one such that your expectations for how the world will turn out, conditional on your performing it, are greater than the expectations conditional on performing some other action. [17]

REILP Approach: The civil and environmental decision-making processes are plagued with uncertain, vague, and incomplete information. Interval linear programming (ILP) is a widely applied mathematical programming method in assisting civil and environmental decision making under uncertainty. Existing ILP decision approach is found to be ineffective in generating operational schemes for practical decision support due to a lack of linkage between decision risk and system return [50]. The REILP (Risk Explicit Interval linear programming) explicitly reflects the tradeoffs between risk and system return for a decision-making problem under an interval type uncertainty environment. REILP approach can obtain crisp solutions at each desired aspiration level or risk tolerance level, which can serve as the basis of directly formulating implementation schemes. Therefore, the solution is straightforward for decision maker to adopt and to make informed decisions with explicit risk-reward tradeoff information. This offers significant advantage over traditional ILP approach, which lacks the capability of providing quantitative relationship between system performance and risk level. The solutions of the REILP are feasible and are guaranteed to be optimal at different system return-risk tolerance level. With the solutions explicitly relating system returns to risk levels, decision makers are in a much more favorable condition to incorporate their own

insight and preferences in the decision-making process to generate effective implementation schemes. [50]

Multiple Criteria Decision Aid (MCDA) Approach: MCDA is a sub-discipline of operations research that explicitly considers multiple criteria in decision-making environments. MCDA structure is important for dealing with typical kind of problems like in the context where a project manager has to take into account several, often contradictory, points of view when he/she is solving a decision problem. Structuring complex problems well and considering multiple criteria explicitly lead to more informed and better decisions. One important change that organizations may experience when using MCDA for strategic decisions, is the use of a value-focused framework to guide the decision-making process. This may help both in aligning the strategic vision of the organization with its strategic objectives, and in better scoping the strategic choices it is considering. [40]

Adaptive Interactive Simulation Framework / Interactive Construction Decision Making Aid (ICDMA): Decision-making in construction projects involves the management of multiple interrelated components such as site layout, critical equipment resources, labor productivity, unexpected events, resource allocation and rescheduling of activities. Uncertainty associated with each of these quantities, and the evolving relationships between them within the constraints of time and space are at the root of complexity in project management. Decisions made in dynamic task environments must consider both immediate impacts and long-term dynamic feedback. A decision strategy is defined as the guideline and

direction that provide the basis for a family of decisions towards achieving a project outcome. [47]

Five strategies that are defined and implemented in ICDMA:

i. A control strategy to reflect baseline conditions. The control strategy aims at completing the project by adhering closely to a baseline decision sequence.
ii. The crash strategy tries to crash the schedule to save time.
iii. The reassign strategy aims to optimize the project by prioritizing activities on the critical path.
iv. The safety strategy aims at reducing the risks due to delayed material delivery by ordering materials ahead.
v. The catch-up strategy tries to catch up with the schedule when it falls behind.

TOPSIS/ VIKOR methods: Contractors play a vital role in the overall performance of a project. Selecting the right contractor for the right project is the most crucial challenge for any construction client. Numerous and often conflicting objectives and alternatives, such as tender price, completion date, and experience, need to be considered. Increased project complexity and higher requirements have demanded the use of multi-criteria decision-making methods for contractor selection. So, two multi-criteria decision methods, the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) and Vlsekriterijumska Optimizacija I Kompromisno Resenje (VIKOR) methods are applied to the selection of a contractor. [43]

a) The TOPSIS method determines a solution with the shortest distance to the ideal solution and the greatest distance from the negative-ideal solution, but it does not consider the relative importance of these distances.

   The TOPSIS procedure consists of the following steps:

   i. Calculate the normalized decision matrix.
   ii. Calculate the weight normalized decision matrix.
   iii. Determine the positive ideal and negative ideal solution.
   iv. Calculate the separation measures, using the n-dimensional Euclidean distance.
   v. Calculate the relative closeness to the ideal solution.
   vi. Rank the preference order.

b) The VIKOR method of compromise ranking determines a compromise solution, closeness to the ideal, providing a maximum "group utility" for the majority, and a minimum of an individual regret for the opponent. This method focuses on ranking and selecting from a set of alternatives in the presence of conflicting criteria. VIKOR is a helpful tool in multi-criteria decision making, particularly in a situation where the decision maker is not able or does not know to express his/her preference at the beginning of system design. [36]

Bayesian theory: Bayesian theory is a probability theory used in decision-making. Bayesian reasoning is a probabilistic extension of logic that enables reasoning with propositions with either a true or false state. Bayesian logic is a branch of logic applied to decision making and inferential statistics that deals with probability

inference: using the knowledge of prior events to predict future events. Bayesian probability theory provides a mathematical framework for performing inference, or reasoning, using probability. [3]

Some algorithms were applied in the past to solve the problem of construction scheduling. Several methods and analytical algorithms such as Critical Path Method (CPM), Integer Programming (IP), Linear Programming (LP) and hybrid IP/LP algorithm were applied to address the problem of construction scheduling optimization [44]. The critical path method (CPM) is a widely used project scheduling algorithm that was developed in the late 1950s [28]. It can be applied to any project with interdependent activities, such as construction, aerospace engineering, software development, industrial manufacturing. To date, CPM is the most commonly used scheduling tool in the construction industry. Fundamentally, however, CPM can only deal with optimization problems with a single objective. CPM is commonly used in conjunction with the Program Evaluation and Review Technique (PERT) [45]. Several efficient approaches for solving the IP problems have been developed such as cutting-plane method, branch and bound method, branch and cut method, and branch and price method [10]. In 1962, Fondahl developed a precedence methodology as an alternative to CPM. The method provides an effective manual process to determine a schedule instead of using a computer-based CPM [20]. Dynamic programming is a mathematical method applicable for solving complex optimization problems that can be broken down into some sub-problems. It is efficient for solving those problems with overlapping sub-problems [13]. Numerous examples of dynamic programming can be found in the construction engineering and management literature. Another method is the heuristic method. Heuristic methods are based on the past

experiences for problem solving. Metaheuristic method, Genetic Algorithm, Ant Colony Optimization and Particle swarm optimization are some of the other algorithms used to solve the problem of construction scheduling optimization.

All the above-mentioned research theories are directed/aimed towards improving decision-making processes of different aspects in construction projects. Although, significant research has been directed towards automating the process of construction schedule using different techniques, algorithm, and tools, industry professionals still prepare construction schedule using software that simply records the information entered, which is a time-consuming process particularly for big projects. Furthermore, preparing such a schedule requires expertise in construction planning and management, which is gained through years of experience. Therefore, there is a need to automate this construction scheduling process to save time and maintain construction expertise over time.

CHAPTER III - PROBLEM DESCRIPTION

Construction of projects face a continuing problem of delayed decision-making in various aspects, especially in the segment of scheduling of tasks/jobs. Developing project schedules is difficult, since it is heavily dependent on project planners' knowledge of work packages, on-the-job-experience, planning capability and oversight [19]. Even to this date, schedules in construction are made manually, which consumes a lot of time. This hints at the need for finding a way to make faster and efficient decisions in making schedules to save time and production cost. Expert systems proved to be very efficient in the process of controlling, scheduling and decision-making. Decision making in construction scheduling plays an important role in finishing the construction projects within an optimum time. Many expert systems related to construction management were developed to solve problems including of construction site layout, construction risk identification, time and cost estimation, and other construction-related issues [8]. As stated earlier, COMIX, BETVAL, BIDEX, AMADEUS, Concrete Mix Designer, and EXPEAR are some of the expert systems used in field of construction [4]. Though there are several expert systems in practice for decision making in different aspects of construction, currently there is no decision support expert system available (that captures the work/practical experiences) to automate the process of scheduling that could assist working professionals and/or students to develop a construction schedule. Hence, this research addresses the problem of manually creating the schedules (which consumes lot of time) for construction tasks by automating the process of making schedules with the use of a computational framework.

3.1 Purpose Statement

The main purpose of this research is to address the issue of manual creation of construction schedules, by developing a computational framework that could help in the process of scheduling jobs/tasks automatically. This framework includes TSP algorithm (to obtain the order of jobs), the Job Shop Problem algorithm (to obtain the order of tasks within the jobs) and possibly Convex Hull algorithm (to differentiate the interior jobs from the exterior jobs).

3.2 Significance of the Study

Upon study and research of several articles on construction scheduling through a period of two years, it is understood that the scheduling of jobs/tasks in a construction project are still carried out manually. This implies the need of an expert system that could automate the process of scheduling in construction. An expert system could also help in the decision-making process while scheduling. My research study focuses on this overlooked part of how expert systems could help construction engineers and students in decision-making process of construction scheduling.

CHAPTER IV - DEVELOPMENTAL OVERVIEW OF THE CONSTRUCTION SCHEDULING DECISION SUPPORT EXPERT SYSTEM

The development of a computational framework for a construction scheduling web-based expert system consisted of multiple applications and programming languages.

4.1 Applications used to develop the Construction Scheduling expert system

BIM Revit: Building Information Modeling (BIM) software application known as Revit is used to design 3D construction models for this project. Revit Architecture is a robust architectural design and documentation software application created by Autodesk for architects and building professionals [6]. The tools and features that make up Revit Architecture are specifically designed to support building information modeling (BIM) workflows.

Dynamo: It is an open source visual programming extension for Revit developed by Autodesk that allows users to control the data, shape the geometry, explore design options, create links between multiple applications and also automate certain processes. Dynamo is a plug in for Revit that can be used to determine the properties of each model in the project. Dynamo extends BIM with the data and logic environment of a graphical algorithm editor. It can also work as a stand-alone application [6]. Dynamo provides users the ability to visually script behavior and script using different textual programming languages. Dynamo will let the users work within a visual programming platform where the elements/nodes are connected in a manner to describe the relation between the nodes and show the sequence of actions. It allows the users to access the external libraries along with its own. It is a flexible and extensible designing tool [14]. The Dynamo plug-in can

be accessed in Revit through the ‘Manage’ tab. When Dynamo plug-in is initiated from Revit, it automatically connects to the active project.

Navisworks: An Autodesk BIM software that is used primarily in construction industries to complement the 3D design packages. Navisworks allows users to open and combine 3D models and navigate through them in real-time. The test case models from Revit are exported to Navisworks (.nwc extension). These exported models are then opened in Navisworks for a 3D visual of the scheduling process.

JESS Expert System: JESS is a rule engine and scripting environment written entirely in Oracle's Java language developed by Sandia National Laboratories [23]. JESS uses an enhanced version of the Rete algorithm to process its rules [7]. An expert system is developed in JESS (a rule-based engine) to calculate the distance matrix and implements the TSP algorithm to produce a sequence of jobs.

The Rete algorithm is a pattern-matching algorithm for implementing production rule systems. Rete algorithm is a very efficient mechanism for solving the difficult many-to-many matching problem [23]. This algorithm has become a base for many rule engines. Rete algorithm is used to determine which of the system's rules should fire based on its data store. The Rete Algorithm is intended to improve the speed of forward-chained rule systems by limiting the effort required to re-compute the conflict set after a rule is fired [23]. Its drawback is that it has high memory space requirements. The Rete algorithm is implemented by building a network of nodes, each of which represents one or more tests found on a rule’s left-hand side (LHS). Facts that are being added to or removed from the working memory are processed by this network of nodes. At the bottom of the network are nodes representing individual rules. When a set of facts filters all the way down to the

bottom of the network and has passed all the tests on the LHS of a particular rule, that rule gets activated [23]. The associated rule may have its right-hand side (RHS) executed if the activation is not invalidated first by the removal of one or more facts from its activation set. Rete eliminates the possibility of any duplicity of the solution by node sharing method. The Rete network is constructed when you compile the rules only once when you start that class and then shared across all requests [11].

There are other rule-based languages such as awk [4], Drools [9], CLIPS [15], OPS5 [9]. However, every language has its limitations such as being able to process only text files like awk [2], some rule-based languages like Drools are only web based and business oriented [15]

A typical rule-based engine has a fixed set of rules while the working memory changes constantly. But in most rule-based engines, working memory is fixed from one rule operation to the next rule operation [20]. Though new facts are collected and old ones are removed continuously, the percentage of facts that change per unit time is generally small. Hence, the implementation of a rule engine becomes very inefficient [20]. This implementation requires keeping a list of the rules and continuously cycling through the list, checking each rule's left-hand-side (LHS) against the working memory and executing the right-hand-side (RHS) of any rules that apply. This is inefficient because most of the tests made on each cycle will have the same results as on the previous cycle. This is a 'rules finding facts' approach and its computational complexity is of the order of $O(RF^P)$, where R is the number of rules, P is the average number of patterns per rule LHS, and F is the number of facts in the working memory [26]. This escalates dramatically as the number of patterns per rule increases [26]. Since JESS uses the Rete algorithm, the inefficiency

described above is lessened by remembering past test results across iterations of the rule loop. Only new facts are tested against any rule in LHSs, to which they are most likely to be relevant [20]. As a result, the computational complexity per iteration drops to something more like O(RFP), or linear in the size of working memory [20]. This use of Rete algorithm by JESS makes it a more efficient rule engine when compared to the others described above.

In this project, the centroids (nodes) of each and every object in the model developed in Revit are extracted. We need to find the optimal (i.e., minimum total cost) sequence of visits to the centroids, such that no centroid is visited more than once. And, so is the objective of the Travelling Salesman Problem. It is to find the shortest possible path from start point to the end point, visiting each and every point exactly once until it returns to the starting point. This is a routing optimization problem. There are numerous techniques and algorithms to approximate the solution of travelling salesman problem, and a few are:

- **Dijkastra's algorithm** is a graph search algorithm that solves the single-source optimal path problem for a graph with nonnegative edge path costs, producing an optimal shortest path tree. But when the network is large, then it becomes inefficient since a lot of computations need to be repeated. The major disadvantage of the algorithm is the fact that it does a blind search, thereby consuming a lot of time waste of necessary resources. Another disadvantage is that it cannot handle negative edges [23, 54].
- **Bellman-Ford algorithm** is a graph search algorithm that finds the shortest path between a given source vertex and all other vertices in the graph. This algorithm can be used on both weighted and unweighted graphs. Like Dijkstra's shortest path

algorithm, the Bellman-Ford algorithm is guaranteed to find the shortest path in a graph. Though it is slower than Dijkstra's algorithm, Bellman-Ford is capable of handling graphs that contain negative edge weights, so it is more versatile. It is worth noting that if there exists a negative cycle in the graph, then there is no shortest path. Going around the negative cycle an infinite number of times would continue to decrease the cost of the path (even though the path length is increasing) [28].

- **A greedy algorithm** is a method for finding an optimal solution to some problem involving a large, homogeneous data structure (such as an array, a tree, or a graph) by starting from an optimal solution to some component or small part of the data structure and extending it, by considering additional components of the data structure one by one, to an optimal global solution [18].
- **Floyd-Warshall algorithm** is a shortest path algorithm for graphs. Like the Bellman-Ford algorithm or the Dijkstra's algorithm, it computes the shortest path in a graph. However, Bellman-Ford and Dijkstra are both single-source, shortest-path algorithms. This means they only compute the shortest path from a single source. Floyd-Warshall, on the other hand, computes the shortest distances between every pair of vertices in the input graph [7].
- **Simulated Annealing** is one of an effective method of finding a good solution for an optimization problem that is bound constrained. Traveling salesman problem can be good example to approximated using this optimization technique. It is used for optimization problems where finding an approximate optimum solution is more important than finding a precise solution in a fixed amount of time [67].

- **A*** is an extension of Dijkstra's algorithm with some characteristics of greedy breadth-first search (BFS) that can use a heuristic to guide itself. It is a technique used in path-finding and graph traversals [15]. The heuristic used to evaluate distances in A* is: f(n) = g(n) + h(n)

  where g(n) represents the exact cost of the path from the starting point to any vertex n,

  h(n) represents the heuristic estimated cost from vertex n to the goal

- **Genetic algorithm** is a method for solving both constrained and unconstrained optimization problems based on the theory of natural selection. Genetic algorithms are exceptional for searching through large and complex data sets. A genetic algorithm differs from a classical, derivative-based, optimization algorithm in two ways [45]:

  1. A genetic algorithm generates a population of points in each iteration, whereas a classical algorithm generates a single point at each iteration.
  2. A genetic algorithm selects the next population by computation using random number generators, whereas a classical algorithm selects the next point by deterministic computation [45].

Since my initial test case size is small and have a smaller number of centroids, I attempted to approximate the solution of shortest path for my research models by using hill-climbing algorithm, since it is the best approximate algorithm for TSP in practice. This algorithm is considered one of the simplest procedures for implementing a heuristic search. Hill climbing is the idea that if you are trying to find the top of the hill then you must go in upward direction from where you are [14, 29]. This heuristic combines the advantages

of both depth-first and breadth-first searches into a single method. Combining the two searches in a single path enables to switch paths whenever some competing path looks more promising than the current one. The HC algorithm may not find the optimal solution to the problem. However, it will give a good solution in reasonable time. A simple HC algorithm has three simple steps as stated in [14, 29]:

Step 1: Evaluate the initial state. If it is a state that satisfies the goal condition (local or global maximum), then stop and return success. Otherwise, make initial state as current state.

Step 2: Loop until the solution state is found or there are no new operators present, which can be applied to current state.

a) Select an operator that is not applied to the current state and apply it to produce a new state.

b) Perform these to evaluate new state

i. If the current state is a goal state, then stop and return success.

ii. If it is better than the current state, then make it current state

iii. If it is not better than the current state, then go to step 2.

Step 3: Exit.

Implement a genetic algorithm to approximate the shortest path for the more complex test cases, with large number of centroids, may yield better results.

4.2 Methodology to develop the construction scheduling decision support expert system

The research of developing a web-based expert system for construction scheduling is a step-by-step approach. The 3D Revit models are acquired, and the required data is extracted from them using the Dynamo plug in Revit. This extracted data contains centroids, model IDs, model types, model categories and learning files for the expert system. Of all the data files that are extracted from Dynamo, certain files (centroids, distance between the centroids) are passed on as the input to the TSP program developed in JESS to obtain a generalized schedule of jobs to be performed. Then, this schedule, expert knowledge that links the jobs to machines (learning files) and, the construction divisions will be given as input to the Job Shop Problem algorithm.

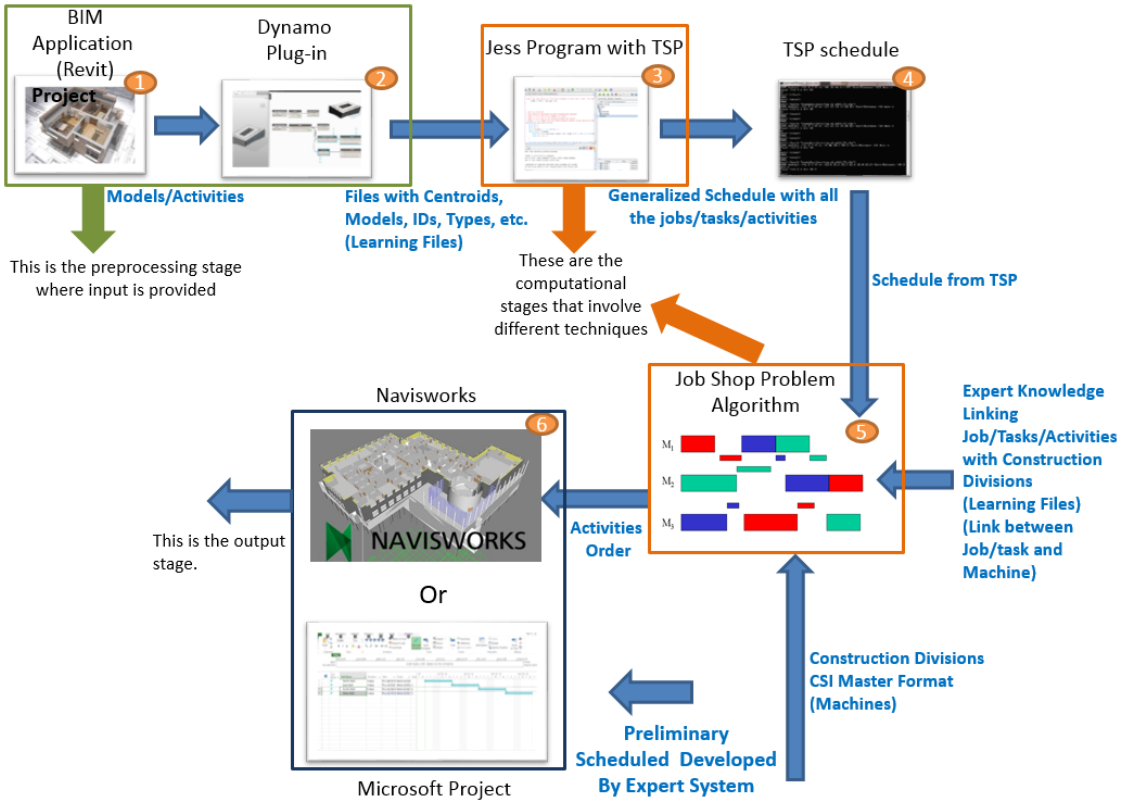

Figure 3 Final framework to obtain the construction scheduling sequence

4.2.1 Designing the test cases Revit models

The test cases for this research are designed in a Building Information Modeling software (Revit). Revit Architecture is a robust architectural design and documentation software application created by Autodesk for architects and building professionals [6]. The tools and features that make up Revit Architecture are specifically designed to support building information modeling (BIM) workflows. BIM is a digital representation of physical and functional characteristics of a facility/project. [11]. Test case 1 consists of three rooms with two mid walls; test case 2 has three rooms and two mid walls with a small closet and test case 3 was six rooms with one having a middle wall (as shown in figure 4). The walls were denoted using cardinal directions. So, the wall facing west was denoted as West Wall (WW), the wall facing north was denoted as North Wall (NW), etc. The middle walls were denoted as Middle Wall (MW) and in the cases where multiple walls facing in one direction a sequential number was added (i.e: MW1 and MW2 in case 1).

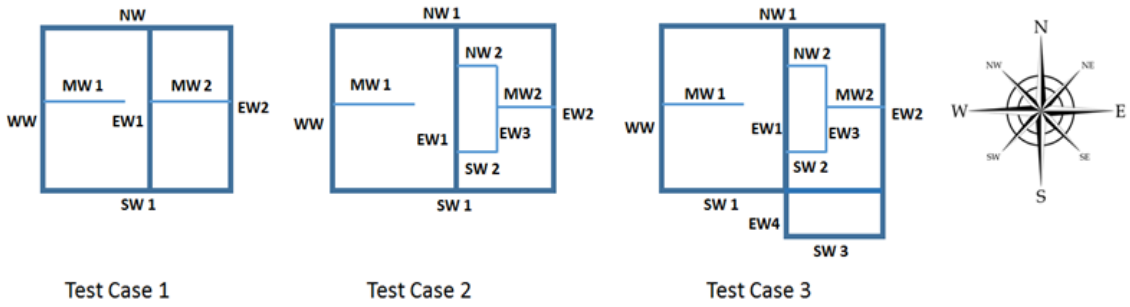

Figure 4 Test Cases for Experiment

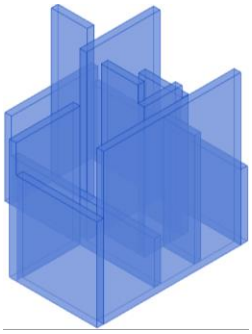

Figure 5 3D model of a Test Case

Development of the scheduling decision support expert system required coding in three different platforms: 1- Dynamo, 2- JESS, 3 - Python. The code developed in the Dynamo platform (Revit plug-in) extracted the centroids of each construction task (represented as a model) from the Revit models and calculated the distances among all the construction tasks. The program developed in JESS used the data from the Dynamo program to prepare a schedule recommendation. The code developed in Python, used the schedule recommendation from JESS, and provided the optimal time taken to finish the tasks of all jobs by assigning the tasks to respective machines and following the constraints. The following is an explanation of the Dynamo code, JESS code and Python code:

4.2.2 Development of code in Dynamo to extract data from the Revit Model

Dynamo code developed for extracting the information from 3D models in Revit needed enhancements to eliminate plane surfaces that do not have a centroid to reduce the error percentage in automated scheduling. This Dynamo code of blocks, finds the information, extracts it, calculates the distances, and saves the required information and parameters into files of desired location.

Step 1- Extract the Centroid of Each Construction Task/Model: Initial step of this entire process is finding the centroids of each construction task/model. This block of code marks the beginning of the entire data collecting process. '*Select Model Elements*' node is the starting point for finding the centroids and distances between them. This node lets you select the model(s) that should be considered to find the centroids (as shown in below figure 6). The '*Solid.Centroid*' node finds the centroids of selected models. The '*Watch'* node displays the output/outcome of the predecessor/connected node.

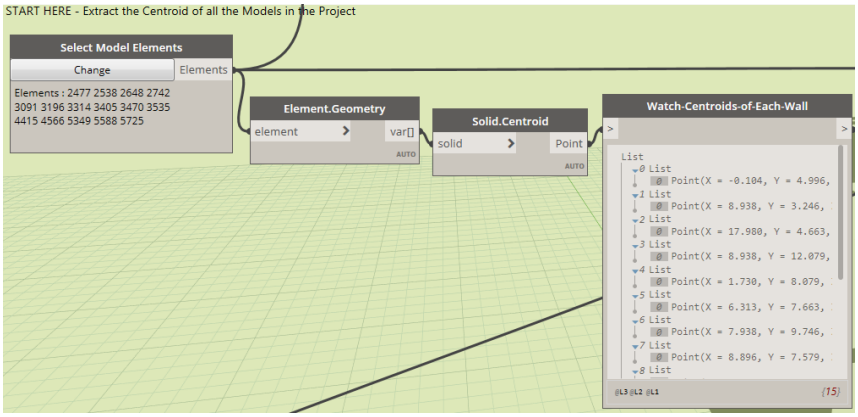

Figure 6 Selecting the models and finding the centroids of each construction task/model

Step 2 - Filter Model Type IDs and Model Names: In the next block of code, the node 'Element.Parameters' captures the parameters (height, width, type, type ID, etc.) of the elements. The node 'String.Contains', 'String.StartsWith' and 'List.FilterByBoolMask' help to filter the list and get the required parameters. 'List.Flatten' node is used to shorten and compress the list. As shown below in figure 7, the parameters 'Type Id' and 'Type' are separated from the initial list of parameters of the elements. Hence, we get the lists of 'Type' and the 'Type Id' separated from the initial list of element parameters.

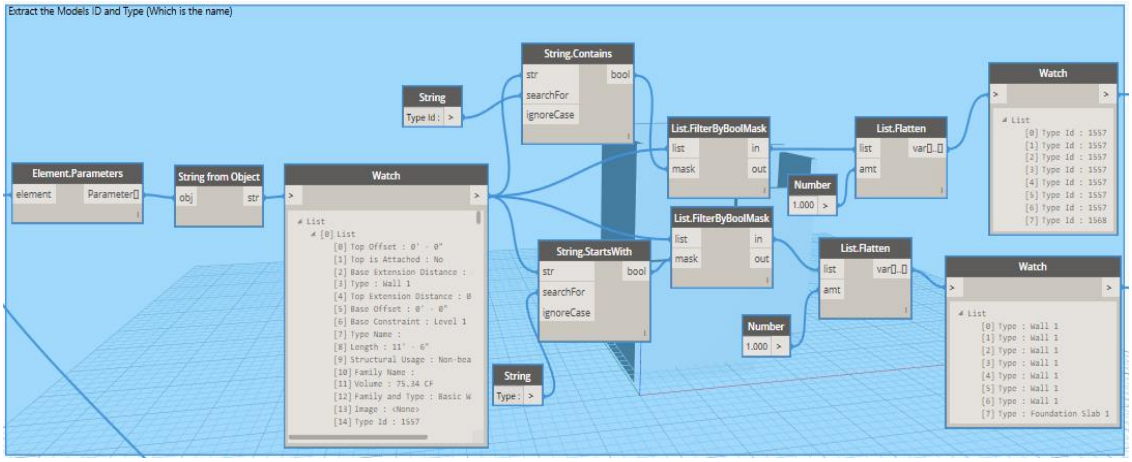

Figure 7 Finding various parameters of the construction models

Step 3 - Extract Model Type ID: Using the 'String.Remove' node, we can delete the unwanted string from the list, in this case the string "Type Id" was eliminated to only save ID number on the file. As shown below in figure 8, the final data is written into a text file using 'CSV.WriteToFile' node. 'File path' node specifies the location of the file.

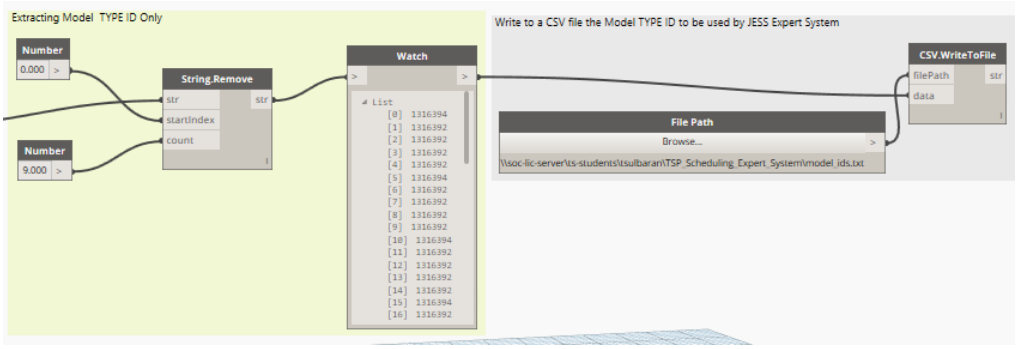

Figure 8 Saving one of the parameters (Type IDs) to a file

Step 4 - Extract Model Name: 'String.Remove' node is used to delete the string "Type" to only save on the file the model name. Figure 9 shows the step to extract model names and write the model names into a text file.

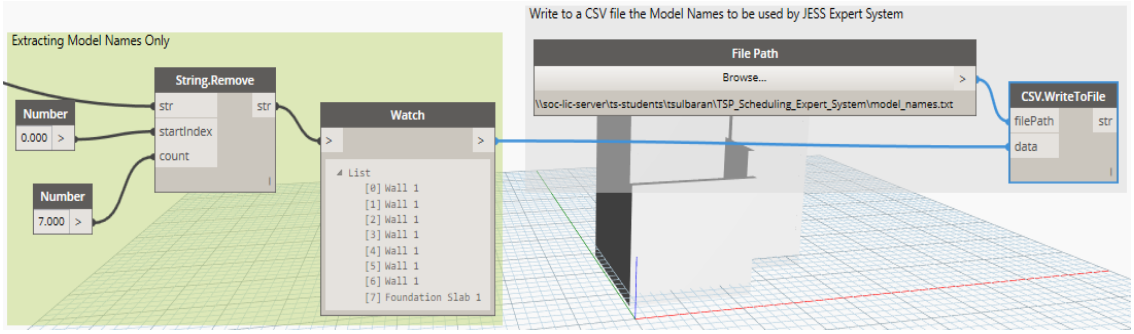

Figure 9 Writing the parameter (model names) to a file

Step 5 - Extract Model Element IDs: As shown in figure 10, ‘Element.Id’ extracts the ID numbers of the elements in the model.

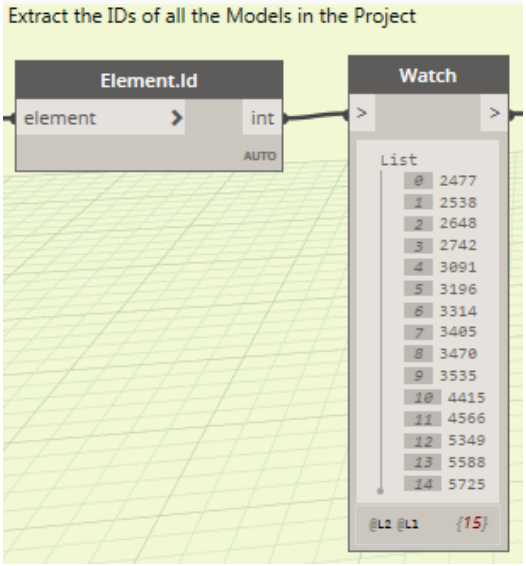

Figure 10 Extracting element IDs of the models

Step 6 - Eliminating Lines, planes and surfaces without centroids: In this box (shown in figure 11), the non-solid models are eliminated from the ID list obtained from step 5. ‘String.Contains’ node filters the input based on the filter parameters. ‘String from Object’ node will give the input to the ‘String.Contains’ node. ‘String’ node gives the parameter to search for in the list. However, before this search is performed, the list obtained in step 1 needs to be cleaned of the null values. The mapped list obtained from a later step (step 7) is used as the input to clear and filter the null values completely from the list. ‘List.FilterByBoolMask’ node filters the list as required.

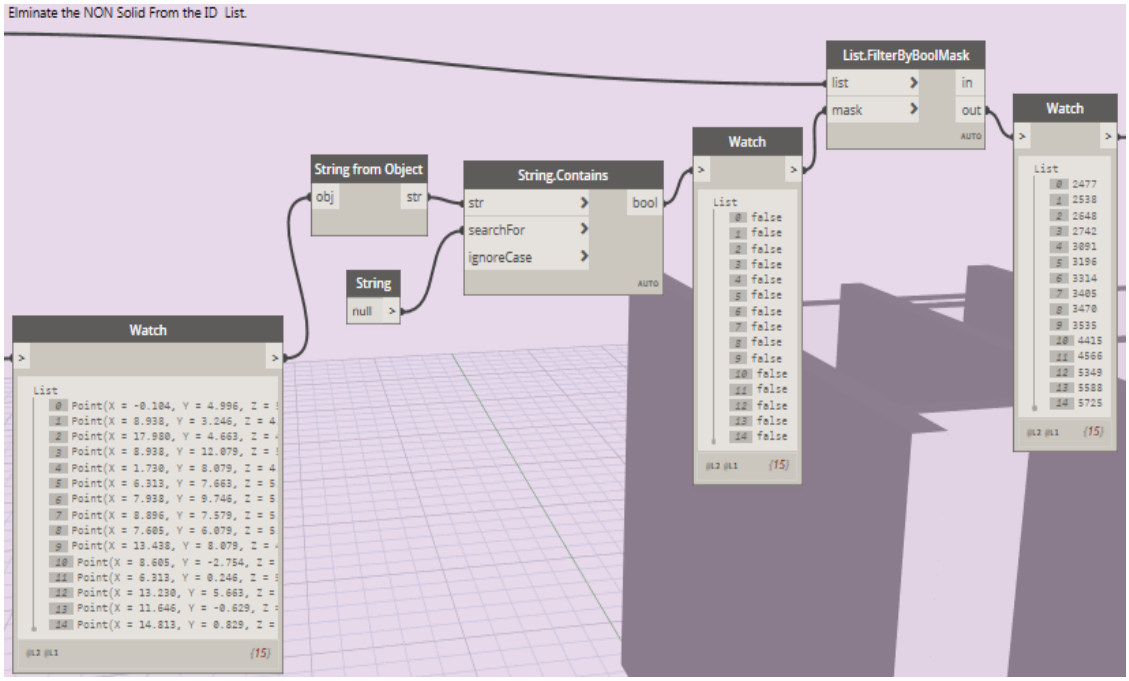

Figure 11 Eliminate non-solids from the ID list

After eliminating, the non-solids form the ID list, the remaining model IDs are written to a file and saved.

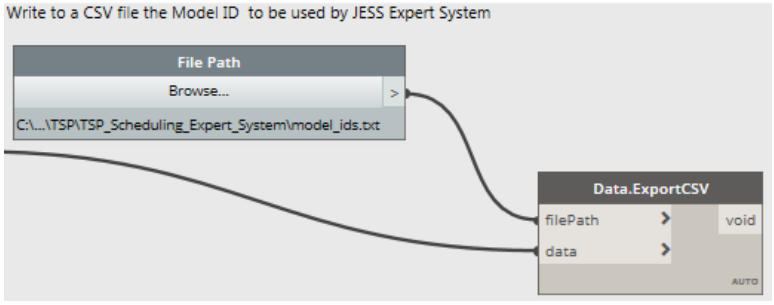

Figure 12 Write model IDs to a file

Step 7 - Clearing any Null values from Centroid List: In this box, the null values are eliminated from the list of centroids without changing the number of values in the list. 'List.Clean' node cleans all the null values from the list and returns an empty list. 'List.Map' node helps to delete the sub lists and maps the list values that are not null to their original list number. The mapped list obtained from this step is used as the input for step 6 to search and delete all the null values from the list.

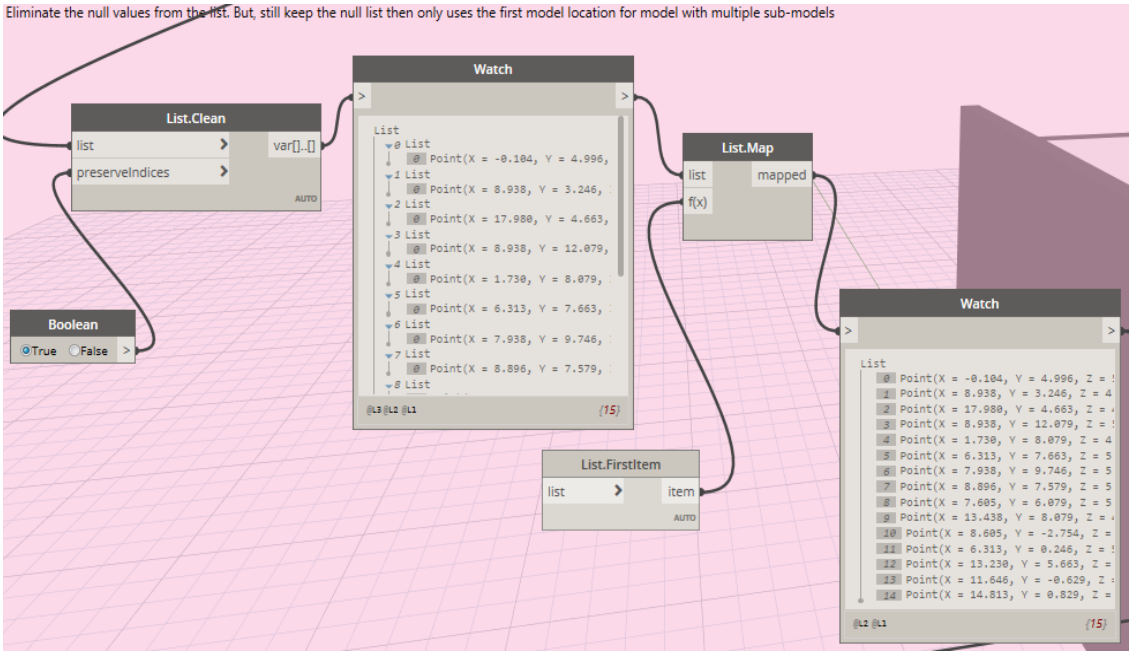

Figure 13 Remove null values from the list

Step 8 - Eliminate the NON-Solids From the centroid List: In this step, we eliminate the non-solid (non-3D) models from the list of the centroids attained from step 1. However, the centroid list has to be first cleaned off all the null values and mapped to the list numbers.

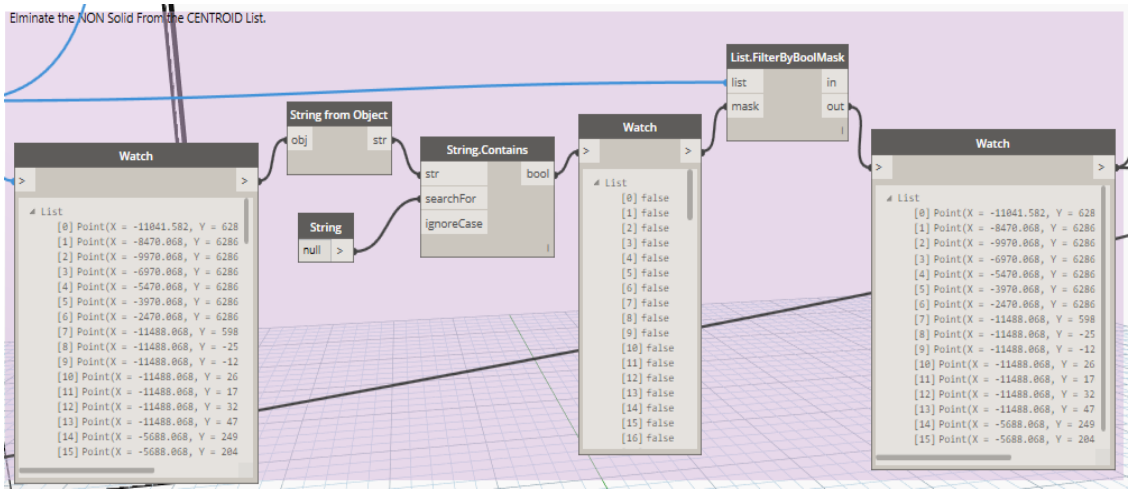

Figure 14 Eliminate the non-solids (null values) from the centroid list

Step 9 - Extract Number of Models in a Project: As shown below in figure 15, the 'List.Count' node helps to count the total number of models that were selected in step 1.

Then the output is written using the 'CSV.WriteToFile' node and saved to a file as specified in the path given in 'File path' node.

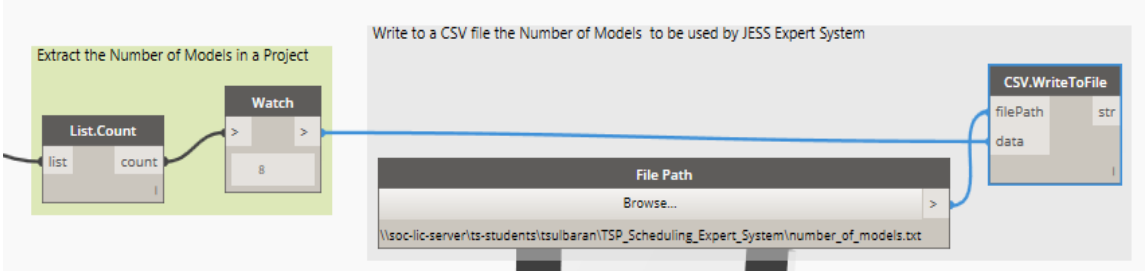

Figure 15 Write number of models into a file

Step 10 - Create all Possible Combinations of the Models to Calculate Distances: Once all the null values and non-solids are removed from the centroid list and mapped to the original list numbering, we need to create a centroid combination pair between all the centroids to calculate distances between them. In the below figure 16, 'List.Combinations' creates all possible lists of centroid combinations for the models to calculate the distances between each pair of centroids.

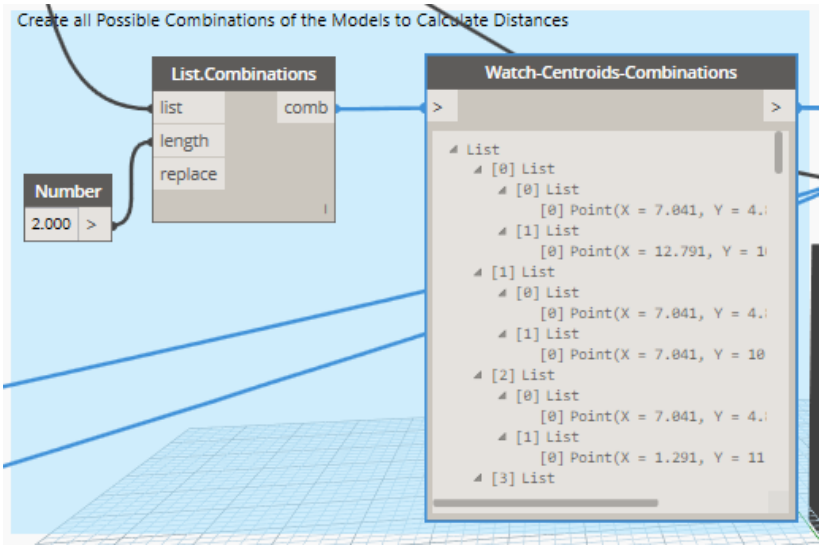

Figure 16 Centroid combinations to calculate distance

Step 11 - Flatten Combination List to get it ready to calculate distance: The centroid combinations are flattened into one single list to ease the distance calculation between them and remove any redundancies. Figure 17 shows the flattened list.

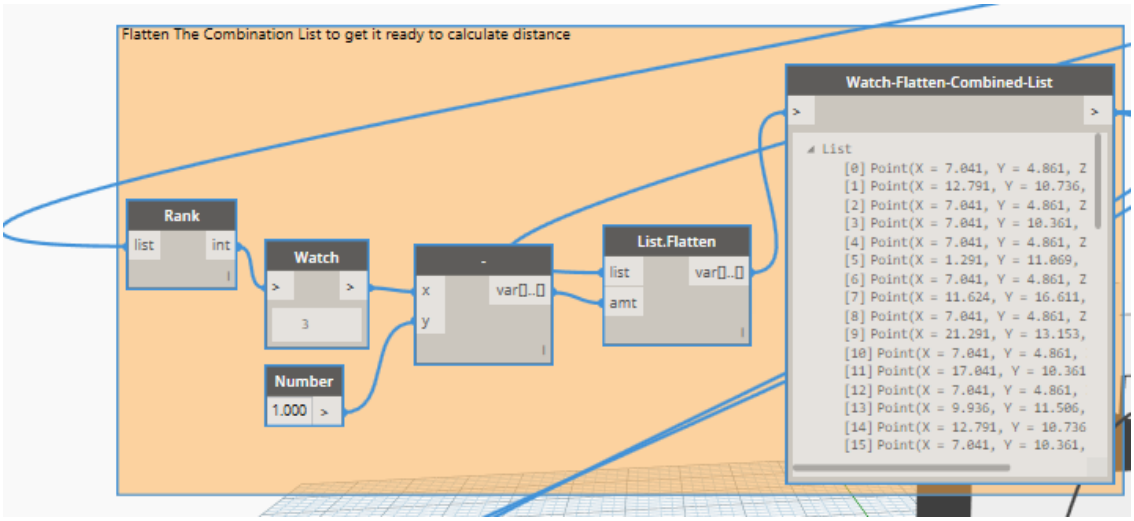

Figure 17 Flatten the centroid combination list

Step 12 - Off set the list by one object to calculate the distance between objects: As shown below in figure 18, the fist centroid is offset from the list of combinations to calculate the distances between the remaining centroids.

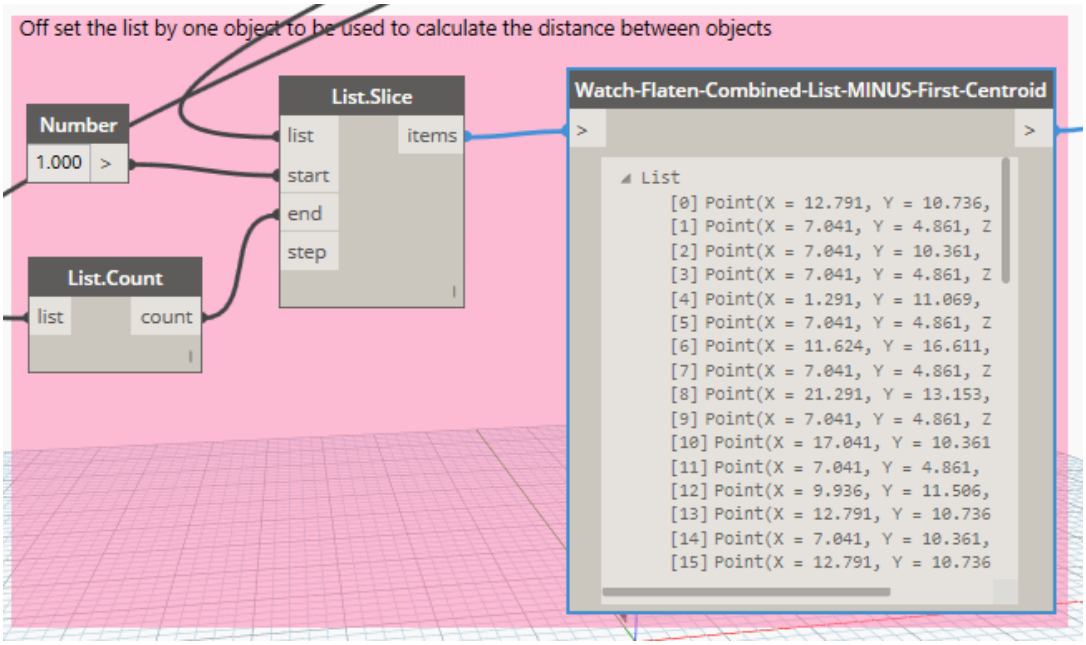

Figure 18 Offset first centroid to calculate the distances

Step 13 - Calculate distance between combinations of centroids: As seen in below figure 19, the node ‘Geometry.DistanceTo’ calculates the distances between the centroids that were calculated earlier.

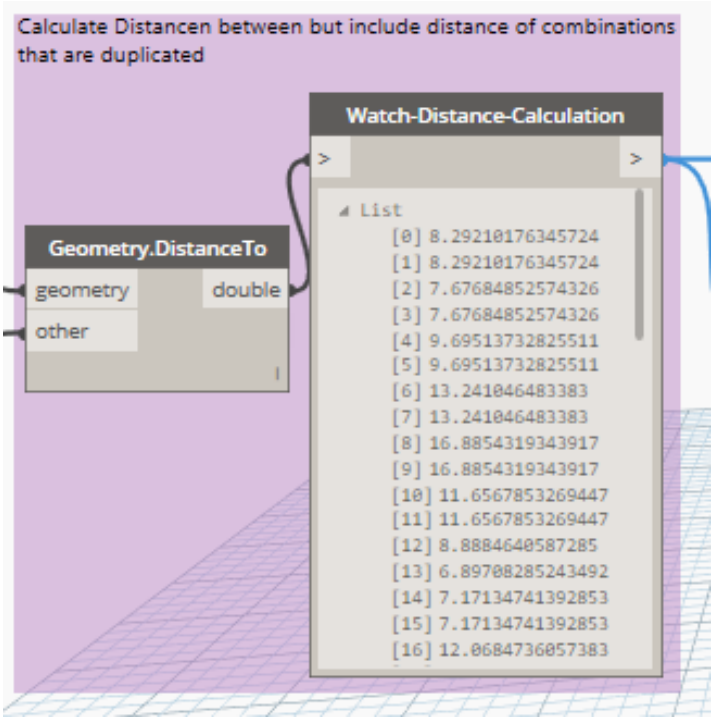

Figure 19 Distance between the centroids (includes duplicates)

Step 14 - Eliminate the distance duplication and generate Final Distance Matrix between Objects: ‘List.Slice’ node helps to delete the redundant/duplicate values. ‘Math.Round’ node can round up the values to a certain decimal place depending on number mentioned in the ‘Number’ node that is attached to the ‘digits’ part of ‘Math.Round’ node. So, the calculated distances between the centroids of all the models are rounded up to two decimal places.

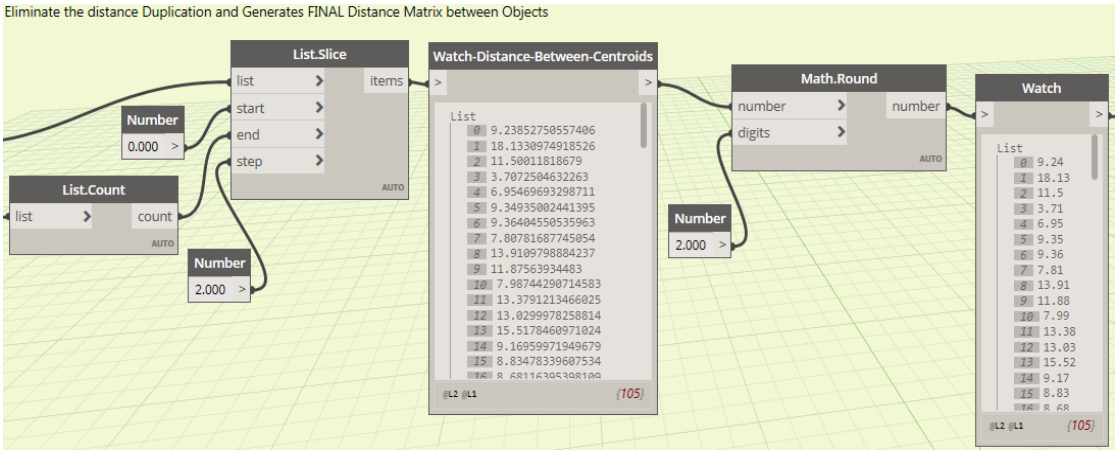

Figure 20 Calculating centroid distances and reducing them to two decimals

The round up values are written and saved into a file. (As shown in figure 21)

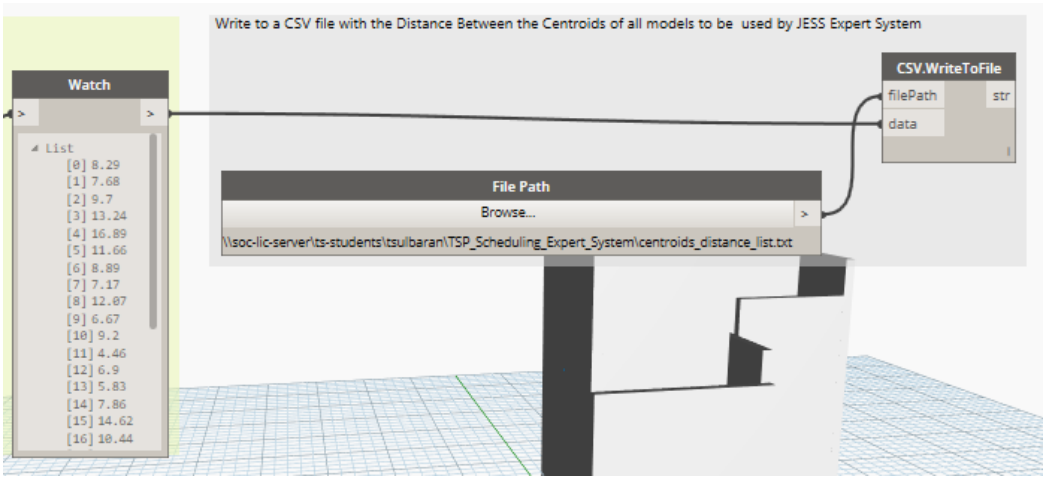

Figure 21 Write the centroids distance list to a file

Figure 22 shows the overall picture of the Dynamo code developed to extract the data from Revit models, and eleminate the unncessary elements to process and generate distance matrix.

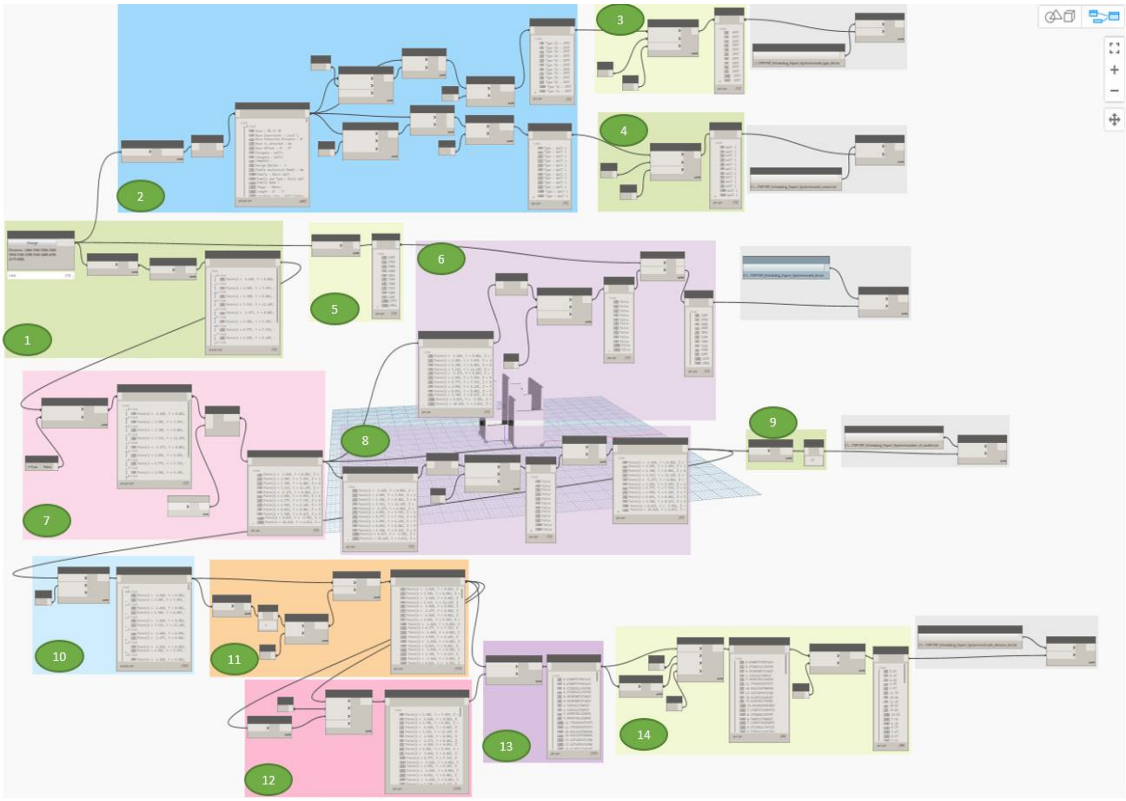

Figure 22 Full view of Dynamo Code to Generate Distance Matrix for Expert System

4.3 Development Overview of the Scheduling Decision Support Expert System in JESS

The Scheduling Decision Support Expert System is developed based on the Travelling Salesman Problem. The TSP was approximated in JESS language implementing functions and rules using the hill-climbing algorithm. It is important to remember that the TSP problem consists of finding a path that passes through a set of cities, in such a way that it must begin and end in the same city and should not visit a city more than once. The activities represented by 3D models in the BIM will take the place of the cities and the schedule will take the place of the path. In construction projects, several kinds of equipment (bulldozers, cranes, heavy lift vehicles etc) need to be transported around the project site for completion of various tasks, which could be expensive and time consuming. Hence, it

is important to approximate TSP and use the generated optimal path to know how the equipment can be navigated/transported while saving cost and time. Below is the detailed explanation of different sections of the JESS code.

The templates ‘deftemplate solution’ and ‘deftemplate mat-dist’ are used to define the distance matrix and the solution. These templates are multi-slot templates, so they can hold a list of values. The ‘deftemplate solution’ has three slots, the solution path, the cost, and a slot that will be used to control the search. Figure 23 shows the JESS code for the two templates. Here ‘path’ corresponds to the sequence of the cities/models of the solution, ‘cost’ corresponds to the total distance of the solution, ‘desc’ corresponds to 'n' for best solution.

```
; Structure of the the Variable for Solution Path
(deftemplate solution
   (multislot path)
   (slot cost (type INTEGER))
   (slot desc)
 )
; Structure of the Variable that contains the Distance between Cities/Models
 (deftemplate model-dist
   (multislot dist)
 )
; Structure of the Variable that contains the names
 (deftemplate model-name
   (multislot name)
 )
```

Figure 23 Templates for Solution Path and Distance between Cities

The function ‘deffunction dist’ is used to assign the number of cities and the distance between the cities. Since the number of nodes and distance between the nodes might not be constant, the values are given in the form on a matrix. Bind function is used to allot the initial values to the variables. ‘(?loopmax)’ is used so the system can evaluate the distances between every possible city. This function transforms the indices to get the

correct position in the list representing the distance matrix. Figure 24 shows the JESS code for this function. This function calculates/locates the distance between two cities from the distance matrix. The inputs are: i is the first city, j is the second city, nciu is the total number of cities in the list, m is a list that contains the distance matrix between cities. The output will be the distance between the two cities.

```
(deffunction dist (?i ?j ?nciu ?m)
     ; this assigns the lowest value between ?i and ?j to ?pm and the highest to ?pn
     (if (< ?i ?j)
        then (bind ?pm ?i) (bind ?pn ?j)
        else (bind ?pm ?j) (bind ?pn ?i)
      )
      ; i is the start of the loop
     ; loopmax is how many times it is repeated which is the lowest value
     (bind ?off 0)
     (bind ?i 1)
     (bind ?loopmax (- ?pm 1))
     (while (<= ?i ?loopmax)
            (bind ?off (+ ?off (- ?nciu ?i)))
            (bind ?i (+ ?i 1))
      )
      ; This line takes position in the first argument in this case (+?off (-?pn ?pm)) from the list ?m
      (nth$ (+ ?off (- ?pn ?pm)) ?m)
)
```

Figure 24 Function to Calculate Distance between Cities

Below figure illustrates the function ‘deffunction cost’, which is used to calculate the accumulating distance from one city to another. This function goes through the list representing the path accumulating the distances. We find the cost/distance of a solution. This function calculates the path accumulating distances. The inputs given are: s (number of cities/models), m (list of the distances between the cities/models). The output generated would be: c (a number with the total distance between cities using the order of cities in s and the distances in m)

```
(deffunction cost (?s ?m)
    (bind ?c 0)
    (bind ?i 1)
    (bind ?loopmax (- (length$ ?s) 1))
    (while (<= ?i ?loopmax)
         (bind ?c (+ ?c (dist (nth$ ?i ?s) (nth$ (+ ?i 1) ?s) (length$ ?s) ?m )))
         (bind ?i (+ ?i 1))
     )
         (bind ?c (+ ?c (dist (nth$ 1 ?s) (nth$ (length$ ?s) ?s) (length$ ?s) ?m)))
)
```

Figure 25 Function to Calculate Accumulated Distance between Cities

The function '*deffunctionload_number_of_models*' is used to assign the number of models. *bind* is used to allot the value(s) to the variable *?location_of_file_with_nukmber_of_models*. As seen in the below figure, the location of the file is given within the double quotes to indicate it is a file at the specified location.

```
; Obtain the number of models in the project
(deffunction load_number_of_models ()
   ;Assign the location of the files needed for the variables to be used later
   (bind ?location_of_file_with_number_of_models "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/number_of_models.txt")
```

Figure 26 Load number of models to the system

In JESS, it is a three-line code to access and open, read or write and close a file saved at a specific location. The code in below figure 27 shows the file specified at above location is opened, 'readline' functions helps to the read the content of the file and 'close' command closes the file. 'assert' function assigns the value in the file to the variable 'num-models'.

```
        ;open file, get the number of models and close file
        (open ?location_of_file_with_number_of_models file1 "r")
        (bind ?number_of_models_from_file (readline file1) )
        (close file1)

        ;define a fact with the number of models
        (assert (num-models ?number_of_models_from_file))
    )
```

Figure 27 Write the number of models to a file

As shown below in figure 28, the '*deffunction convert_distance_file*' is used to the load the number of models, the distances between the centroids, and the distance matrix into the program.

```
(deffunction convert_distance_file ()
   ;Assign the location of the files needed to variables to be used later
   (bind ?location_of_file_with_distance_matrix "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/Distance_Matrix.clp")
   (bind ?location_of_file_with_distance_from_dynamo "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/centroids_distance_list.txt")
   (bind ?location_of_file_with_number_of_models "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/number_of_models.txt")
```

Figure 28 Load models, centroid distances and distance matrix

The data in the file should be in certain format (syntax) for JESS to read the content of it. Hence, the file needs to be opened, modified and closed with JESS commands. The first step would be to read the number of models saved in the file with number of models. Next, to open the file with distance matrix and write '(bind ?m (create$' in the file and close it. The final step would be to open the distance matrix file again and append it. The centroid values from the dynamo file are written in to the distance matrix file and the file is closed. Figure 29 illustrates these steps.

```
;open file, get the number of models and close file
(open ?location_of_file_with_number_of_models file1 "r")
(bind ?nm (readline file1) )
(close file1)

; open the the file that will create the variable with the distance matrix
; and creates the beginning of the file
(open ?location_of_file_with_distance_matrix  file1 "w")
(printout file1 "(bind ?m (create$ "  crlf)
(close file1)

; open the file with the distance between models, get each one of the
; distances and add them to the file that creates the distance matrix variable
(open ?location_of_file_with_distance_matrix file1 "a")
(open ?location_of_file_with_distance_from_dynamo file2 "r")
```

Figure 29 Append the file to JESS syntax

The code in the below figure 30, shows the loop that is to be executed to calculate the number of the values to be present in the distance matrix based on the number of models.

```
    ; the number of distances in the matrix is defined by the equation below
    (bind ?loopmax     (- (* 0.5 (* ?nm ?nm))   (* .5 ?nm))    )
    (bind ?i 1)
    (while (< ?i ?loopmax)
        ; read the distance in each line of the file
        (printout file1 (readline file2) " ")
        (bind ?i (+ ?i 1))
     )
    (printout file1 (readline file2))
    (printout file1 "))"  crlf)
    (close file2)
    (close file1)
)
```

Figure 30 Loop to define the distance matrix

'defrule init' initializes the search. This rule will only run once at the beginning. It is responsible for creating the list representing the distance matrix for the n cities. It will present the initial solution that will visit the cities in sequential order.

```
(defrule init
   (num-models ?x)
    =>
    (printout t "THIS IS X : "?x crlf)
(convert_distance_file)
(batch "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/Distance_Matrix.clp")
```

Figure 31 Rule to initialize the search

For every iteration, all possible city interchanges are generated and the best one is kept. To determine all possible exchanges the JESS inference engine was used to perform the calculations itself. Only an auxiliary fact is needed to control the generation of combinations. This fact was called 'pos' and there are as many facts as positions and each one has one of the possible values. To generate all pairs, we merely needed a condition like '(pos ?i) (pos ?j)'. The constraint '(pos ?i) (pos ?j&:(> ?j ?i))' was added so that the pairs are not duplicated. The algorithm will stop when from the current solution no better solution can be generated as shown in figure 32 below. This algorithm works based on the following three steps:

1. Initializing the search by creating an initial solution.

2. Generating all the descendants by swapping the cities.

3. Choosing the best city and updating the solution or stopping if there is no new solution.

In the below code, the content of the Distance Matrix in the variable *?m* is converted to a fact of the multislot *model-dist*. This part of code creates the facts "pos" of the first run with the position 1 through the number of cities/models.

```
    (assert (model-dist (dist ?m)))
    ; Control fact for the swaps where ?x is the number of cities/models in the project
    (bind ?loopmax ?x)
    (bind ?i 1)
    (while (<= ?i ?loopmax)
        (assert (pos ?i))
        (bind ?i (+ ?i 1))
     )
    ; Initial solution (Cities in sequential order)
       (bind ?s (create$))
      (bind ?loopmax ?x)
      (bind ?i 1)
      (while (<= ?i ?loopmax)
          (bind ?s (insert$ ?s ?i ?i))
          (bind ?i (+ ?i 1))
       )
; Calculate Initial sequence ?s with it cost for a distance matrix ?m of the solution
     (assert (solution (path ?s) (cost (cost ?s ?m)) (desc n)))
  (printout t "INITIAL SOLUTION : " ?s crlf)
)
```

Figure 32 Distance Value Insert and Swap in the Expert System

The rule in the Figure performs the generation of each solution iteration. It instantiates the rule to create a solution that will later be compare against the best solution already obtained.

```
; This rule performs the generation of each iteration of all possible swaps
(defrule HC-step1
    (pos ?i)
    (pos ?j&:(> ?j ?i))
    (solution (path $?s) (cost ?c) (desc n))
    (model-dist (dist $?m))
     =>
    ; assign to ?vi the value in the location ?i of the path ?s
    (bind ?vi (nth$ ?i ?s))
    ; assign to ?vj the value in the location ?j of the path ?s
    (bind ?vj (nth$ ?j ?s))
    (printout t  "s :" ?s " | i :" ?i " vi :" ?vi  " | j :" ?j " vj :" ?vj crlf )

    ; delete from the path ?s the item ?i and assign the remaining to ?sm
    (bind ?sm (delete$ ?s ?i ?i))
    ; insert in the path ?sm
    (bind ?sm (insert$ ?sm ?i ?vj))
    (bind ?sm (delete$ ?sm ?j ?j))
    (bind ?sm (insert$ ?sm ?j ?vi))
    (printout t  "sm :" ?sm  crlf crlf)

    ; calculate the cost of solution/path ?sm based on distance matrix in ?m
    (bind ?nc (cost ?sm ?m))

    ; creates a new fact with the solution ?sm with a cost ?nc and desc s
    (assert (solution (path ?sm) (cost ?nc) (desc s)))
    (printout t "Another Solution->" ?sm   " Cost/Distance :" ?nc " s" crlf)
 )
```

Figure 33 Expert System Solution Finder

After finding a solution, it must be compared against the best solution already obtained. Therefore, the rule showed in figure 34 compares the newfound solution against the best solution that the expert system has. This comparing rule has a '(declare (salience -10))' to give it a lower priority than the rule that finds the solutions (shown in figure 34). This tells the expert system to execute the solution comparison after each solution is found.

```
(defrule HC-step2
    (declare (salience -10))
    (solution (path $?s) (cost ?c) (desc s))
    (solution (cost ?oc) (desc s))
    (test (<= ?c ?oc))
     ?solact <- (solution (cost ?cs&:(< ?c ?cs)) (desc n))
    =>
    (modify ?solact (path ?s) (cost ?c))
 )
```

Figure 34 Rule comparing the best-known solution to the new solution

The main goal is to find a solution that has the minimum cost and that has a cost better than the current solution. After all possible solutions have been determined; the optimal solution (shortest path) is presented by the rule shown in figure 35. This rule has a very low salience number which make it wait until all other rules of the expert system have been applied and completed.

```
(defrule HC-is-end
   (declare (salience -20))
   (solution (path $?s) (cost ?c) (desc n))
   =>
   (printout t "This is the best solution on:" (time)  crlf)
   (printout t "Final ->" ?s ": " ?c crlf)
)
```

Figure 35 Rule that shows most optimal solution

The rule below in figure 36 defines the code for final output. This rule will be executed at the very end since it has the least salience. This is the rule to print the best solution ?s that have least cost/distance ?c and save it to a file. This rule is given a lower salience number to make the rule executed in the very end. Thus, this is the last rule executed in this program

```
(defrule Outputfile
    (declare (salience -30))
    (solution (path $?s) (cost ?c) (desc n))
    =>
    ;Assign the location of the files needed to variables to be used later
    (bind ?location_of_file_with_model_names "Z:/tsulbaran/TSP_Scheduling_Expert_System/model_names.txt")
    (bind ?location_of_file_with_model_ids "Z:/tsulbaran/TSP_Scheduling_Expert_System/model_ids.txt")
    (bind ?location_of_file_for_output_for_naviswork "Z:/tsulbaran/TSP_Scheduling_Expert_System/output_for_naviswork.csv")
    (bind ?location_of_file_with_number_of_models "Z:/tsulbaran/TSP_Scheduling_Expert_System/number_of_models.txt")
```

Figure 36 Assign the files to the variables and order to the rule

The below lines of code in figure 37, reads the names of models and their unique model IDs.

```
;Read the file with the models names and assign the names to the variable ?model name
(printout t ?location_of_file_with_model_names crlf)
(open ?location_of_file_with_model_names file1 "r")
(open ?location_of_file_with_model_ids file2 "r")
(bind ?model-name (create$))
(bind ?model-id (create$))
```

Figure 37 Reads model names and their model IDs

As shown in figure 38, the number of models in the project are read from this part of the program.

```
;open file, get the number of models and close file
(open ?location_of_file_with_number_of_models file3 "r")
(bind ?number_of_models (readline file3) )
(close file3)
```

Figure 38 Read number of models in the project

The lines of code in figure 39 shows the model names and their unique model IDs being matched before saved to a file.

```
;HERE NUMBER OF MODELS XXXX
(bind ?loopmax ?number_of_models)
(bind ?i 1)
(while (<= ?i ?loopmax)
    (bind ?model-name  (insert$ ?model-name (+ (length$ ?model-name) 1)  (readline file1) ) )
    (bind ?model-id  (insert$ ?model-id(+ (length$ ?model-id) 1)  (readline file2) ) )
    ;(printout t "model-name: " ?model-name " | model-id:" ?model-id  crlf)
    (bind ?i (+ ?i 1))
)
(close file1)
(close file2)
```

Figure 39 Matches model names with their unique model IDs

The below line of codes open an excel file and write the following information in to the cells. And then, it goes through the complete list of jobs and link the name of the model with the respective model number in the order of the best solution. It also displays the date and time scheduled for that particular job.

```
;Goto through the complete list and link the name of the model with the model number in the order of the best solution
(open ?location_of_file_for_output_for_naviswork file1 "w")
(printout file1 "ID,Active,Task Mode,Task Name,Duration,Planned Start Date,Planned End Date,Predecessors,Outline Level,Tast Type ,Revit ID" crlf)
(close file1)
(open ?location_of_file_for_output_for_naviswork file1 "a")
(bind ?i 1)
(while (<= ?i ?loopmax)
    ;Shows ID,  Active, Task
    (printout t ?i ",Yes,Manually Scheduled,")
    ;Shows Task Name
    (printout t  (nth$ (nth$ ?i ?s) ?model-name))
    ;Shows Duration, Planned Start Date, Planned End Date, Predecessors, Outline Level , Task Type
    (printout t ",1 day,11/" ?i "/2018 8:00,11/" (+ ?i 1) "/2018 17:00,"   (- ?i 1) ", 1,Construct," )
    ;Shows Revit ID
    (printout t  (nth$ (nth$ ?i ?s) ?model-id) )
    ;(printout t "," (nth$ ?i ?s)
    (printout t crlf)
    (printout file1 ?i ",Yes,Manually Scheduled,")
    (printout file1 (nth$ (nth$ ?i ?s) ?model-name) )
    (printout file1 ",1 day,11/" ?i "/2018 8:00,11/" (+ ?i 1) "/2018 17:00,"   (- ?i 1)  ", 1,Construct," )
    (printout file1  (nth$ (nth$ ?i ?s) ?model-id))
    ;(printout file1 "," (nth$ ?i ?s)
    (printout file1 crlf)
    (bind ?i (+ ?i 1))
)
(close file1)
```

Figure 40 Link name and model number in the order of the best solution

The figure 41 below shows the code to save the best found solution to a text file, that will later be accessed by the Python code to schedule and sequence the tasks.

```
    (open "C:/Users/ferozahmed/Desktop/TSP/TSP_Scheduling_Expert_System/output.txt" file2 "a")
    (printout file2 "This is the best solution on:" (time)  crlf)
    (printout file2 "Final ->" ?s ": " ?c crlf)
    (close file2)
    (printout t "Output file created ->" ?s " on " (time) crlf)
)

(reset)
```

Figure 41 Write the best solution found to a text file.

The below code loads total number of models to before finding the optimal schedule.

```
(load_number_of_models)
(run)
(facts)
```

Figure 42 Loading number of models

4.4 Development Overview of the Scheduling Decision Support Expert System in Python incorporating Job Shop Scheduling algorithm

The main objective of the job shop problem algorithm is to minimize the total time taken from the start of first task of the first job until the end time of the final task of last job. Following is the brief explanation of the Python code used as part of this research. Majority of the code is obtained from the Job Shop Problem developed by Google developers, and was edited and modified as per the requirements of this research. In the initial step, the required OR-Tools are imported, and the model of the problem is defined. The below figure 43 shows the lines of code to import the Python wrapper and solver from OR-Tools. OR-Tools is an open source software developed for solving optimization problems [45].

```
from __future__ import print_function

import collections

# Import Python wrapper for or-tools CP-SAT solver.
from ortools.sat.python import cp_model


def main(read_jobs):
    """Minimal jobshop problem."""
    # Create the model.
    model = cp_model.CpModel()
```

Figure 43 Import tools and solver & declare the model

In the lines of code shown in figure 44, the data required for the problem to be solved is defined. First, the number of jobs is defined in the main function, followed by the machine sequence required for each job. Once the input data is defined, the data is validated comparing the values given. After validating the input for machines, the system prompts for processing times inputs.

```
jobs_data = [] # task = (machine_id, processing_time)

for job in read_jobs:
    machine_ele = []
    user_value = input("Provide machine sequence for processing Job " + str(job) + " tasks: ")
    #convert string into a list
    machine_ele = [int(x) - 1 for x in user_value.split()]

    #validate input
    for val in machine_ele:
        if val < 0 or val > 3:
            print("Only 4 machines are available. Value should be in between 1 and 4")
            return

    process_ele = []
    element = 4
    user_value = input("Enter the processing times for job " + str(job) + " tasks: ")
    #convert string into a list
    process_ele = [int(x) for x in user_value.split()]

    #check if input is valid
    if len(machine_ele) != len(process_ele):
        print("Available machine mismatch. Please recheck")
        return

    #create job data
    temp_list = []
    for i in range(0, len(machine_ele)):
        temp_list.append((machine_ele[i], process_ele[i]))
    jobs_data.append(temp_list)

machines_count = 1 + max(task[0] for job in jobs_data for task in job)
all_machines = range(machines_count)
jobs_count = len(jobs_data)
all_jobs = range(jobs_count)
```

Figure 44 Define the data of the problem

Figure 45 shows the lines of code that defines the variables of the problem. For each individual job and task, this program uses the solver's **NewIntVar** method to create the following variables:

- **start_var**: Start time of the task.
- **end_var**: End time of the task.

The upper bound for **start_var** and **end_var** is horizon, the sum of the processing times for all tasks in all jobs. horizon is sufficiently large to complete all tasks for the following reason: if you schedule the tasks in non-overlapping time intervals (a non-optimal

solution), the total length of the schedule is exactly horizon. So the duration of the optimal solution cannot be any greater than horizon [45].

Next, the program uses the **NewIntervalVar** method to create an interval variable—whose value is a variable time interval—for the task [45]. The inputs to **NewIntervalVar** are:

- start_var: Variable for the start time of the task.
- duration: Length of the time interval for the task.
- end_var: Variable for the end time of the task.
- 'interval_%i_%i' % (job, task_id)): Name for the interval variable.

```
task_type = collections.namedtuple('task_type', 'start end interval')
assigned_task_type = collections.namedtuple('assigned_task_type',
                                            'start job index')

# Create jobs.
all_tasks = {}
for job in all_jobs:
    for task_id, task in enumerate(jobs_data[job]):
        start_var = model.NewIntVar(0, horizon,
                                    'start_%i_%i' % (job, task_id))
        duration = task[1]
        end_var = model.NewIntVar(0, horizon, 'end_%i_%i' % (job, task_id))
        interval_var = model.NewIntervalVar(
            start_var, duration, end_var, 'interval_%i_%i' % (job, task_id))
        all_tasks[job, task_id] = task_type(
            start=start_var, end=end_var, interval=interval_var)
```

Figure 45 Define the variables of the problem

The following lines of code below in figure 46 defines the disjunctive and conjunctive constarints of the problem. The program uses the solver's AddNoOverlap method to create the no overlap constraints, which prevent tasks for the same machine from overlapping in time. Next, the program adds the precedence constraints, which prevent consecutive tasks for the same job from overlapping in time [45].

```
# Add disjunctive constraints.
for machine in all_machines:
    intervals = []
    for job in all_jobs:
        for task_id, task in enumerate(jobs_data[job]):
            if task[0] == machine:
                intervals.append(all_tasks[job, task_id].interval)
    model.AddNoOverlap(intervals)

# Add conjunctive constraints.
for i in range(0, len(all_jobs)):
    for task_id in range(0, len(jobs_data[all_jobs[i]]) - 1):
        model.Add(all_tasks[all_jobs[i], task_id +
                            1].start >= all_tasks[all_jobs[i], task_id].end)

    if i < len(all_jobs) - 1:
      model.Add(all_tasks[all_jobs[i + 1], 0].start >= all_tasks[all_jobs[i], 0].end)
```

Figure 46 Constraints

The below code in figure 47 defines the objective of the problem. That is, to minimize the time taken to finish all the tasks of all jobs.

```
# Makespan objective.
obj_var = model.NewIntVar(0, horizon, 'makespan')
model.AddMaxEquality(
    obj_var,
    [all_tasks[(job, len(jobs_data[job]) - 1)].end for job in all_jobs])
model.Minimize(obj_var)
```

Figure 47 Define the objective

The following code in the below figure 48 declares constraint programming solver and calls the solver.

```
# Solve model.
solver = cp_model.CpSolver()
status = solver.Solve(model)
```

Figure 48 Declare the solver

```
# Print out makespan.
print('Optimal Schedule Length: %i' % solver.ObjectiveValue())
print()

# Create one list of assigned tasks per machine.
assigned_jobs = [[] for _ in all_machines]
for job in all_jobs:
    for task_id, task in enumerate(jobs_data[job]):
        machine = task[0]
        assigned_jobs[machine].append(
            assigned_task_type(
                start=solver.Value(all_tasks[job, task_id].start),
                job=job,
                index=task_id))

disp_col_width = 10
sol_line = ''
sol_line_tasks = ''
print('Optimal Schedule', '\n')
for machine in all_machines:
    # Sort by starting time.
    assigned_jobs[machine].sort()
    machine_name = ''
    if machine == 0:
      machine_name = "Carpenter (060100): "
    elif machine == 1:
      machine_name = "Insulator (072100): "
    elif machine == 2:
      machine_name = "Electrician (260500): "
    elif machine == 3:
      machine_name = "Finisher (060600): "

    sol_line += machine_name + ': '
    sol_line_tasks += machine_name + ': '

    for assigned_task in assigned_jobs[machine]:
        name = 'job_%i_%i' % (read_jobs[assigned_task.job], assigned_task.index + 1)
        # Add spaces to output to align columns.
        sol_line_tasks += name + ' ' * (disp_col_width - len(name))
        start = assigned_task.start
        duration = jobs_data[assigned_task.job][assigned_task.index][1]
        sol_tmp = '[%i,%i]' % (start, start + duration)
        # Add spaces to output to align columns.
        sol_line += sol_tmp + ' ' * (disp_col_width - len(sol_tmp))
    sol_line += '\n'
    sol_line_tasks += '\n'

print(sol_line_tasks)
print('Time Intervals of Tasks\n')
print(sol_line)
```

Figure 49 Displays the result

```
if __name__ == '__main__':
    #read file path. Where file is kept
    file_path = input("Enter jobs file path: ")
    infile = open(file_path, "r")
    #read file
    lines = infile.read().strip().split("\n")
    #for each line do processing
    for line in lines:
        if len(line) > 1:
            #convert line into list of job sequence
            all_jobs = [int(x) for x in line.split()]
            print("Processing for job sequence: " + line)
            main(all_jobs)
            print("\n\n---------------------------------------\n\n")
    infile.close()
```

Figure 50 Main function

To explain the constraints of the problem, we use the following example. One of the test case models used for this research has 12 walls but different tasks for each wall with different processing times. Below figure 51 shows this test case model, which has six interior wall and six exterior walls. The walls are denoted using cardinal directions. So, the wall facing west is denoted as West Wall (WW), the wall facing north is denoted as North Wall (NW), etc. The middle wall is denoted as Middle Wall (MW) and for multiple walls facing in one direction, a sequential number is added (i.e: SW1, SW2 and SW3). On the left, the walls are numbered for an easy undesrtanding of the schedule.

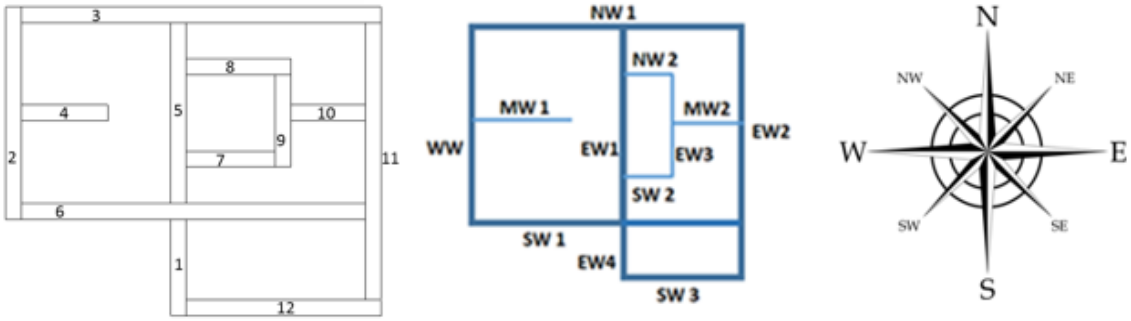

Figure 51 Test case model

This model has twelve walls, out of which six walls (4, 5, 7, 8, 9 and 10) are partition/interior walls and six walls (1, 2, 3, 6, 11, and 12) are structural/exterior walls.

Here, jobs 1, 2, 3, 6, 11, and 12 refer to construction of a structural wall and jobs 4, 5, 7, 8, 9 and 10 refer to construction of a partition wall. Structural wall jobs have 4 specific sequential tasks to be performed, and partition wall jobs have 3-4 sequential tasks to be performed. Each task for a job is given by (**m**, **p**) where **m** is the machine number that the task must be processed on and **p** is the processing time of that particular task. Below is the order of jobs and tasks for each job.

job 1 = [(1, 3), (2, 2), (3, 2), (4, 2)]
job 2 = [(1, 2), (2, 3), (3, 3), (4, 2)]
job 3 = [(1, 2), (2, 1), (3, 1), (4, 2)]
job 4 = [(1, 3), (2, 2), (5, 2)]
job 5 = [(1, 2), (2, 1), (3, 1), (5, 2)]
job 6 = [(1, 3), (2, 2), (3, 2), (4, 2)]
job 7 = [(1, 1), (2, 2), (5, 1)]
job 8 = [(1, 1), (2, 1), (5, 1)]
job 9 = [(1, 2), (2, 1), (4, 2)]
job 10 = [(1, 1), (3, 1), (4, 2)]
job 11 = [(1, 2), (2, 1), (3, 1), (4,3)]
job 12 = [(1, 2), (2, 2), (3, 1), (4,1)]

In this test case, job 1 has four tasks. The first task, (1, 3), must be performed on machine 1 in 3 units of time. Likewise, the second task, (2, 2), must be performed on machine 2 in 2 units of time and so on and so forth.

For jobs 1, 2, 3, 6, 11 and 12 the tasks are in the sequence of:

Task 1→Task 2→Task 3→Task 4

Task 1 - putting up the wood (front side), to be performed by M1.

Task 2 - providing insulation, to be performed by M2.

Task 3 - doing wiring for the wall, to be performed by M3.

Task 4 - putting up the plywood for the wall (back side), to be performed by M4.

For jobs 4, 5, 7, 8, 9, and 10, there are three or four sequential tasks. Machine 5 is an assembler for the tasks of interior jobs.

Let us assume task *t* (*i, j*) denotes j^{th} task in the sequence for job *i*. So, for example task *t* (1, 3) denotes the *third* task for job *1*, which corresponds to the pair (3, 2) for job 1 in this problem.

Next, let $T_{i,j}$ be the start time for task *t* (*i, j*). $T_{i,j}$, is the variable in this problem. Finding a solution for this problem involves finding the value of this variable that meets the requirements of the problem.

Conjunctive constraints: For any two consecutive tasks in the same job, the first must be completed before the second can be started. In the above example, task *t (1, 1)* and task *t (1, 2)* corresponding to the pairs (1, 3) and (3, 2) for job 1 are consecutive tasks for job *1*. Since the processing time for task *t (1, 1)* is 3, the start time for task *t (1, 2)* must be at least 3 units of time after task 1 has started. As a result, you get the following constraint:

$$\boldsymbol{T_{1,2} + 2 \leq T_{1,3}}$$

Disjunctive constraints: A machine cannot work on two tasks at the same time. For example, task *t (1, 3)* and task *t (2, 3)* are both processed on machine 3. The constraint depends on which task is scheduled first and which one is scheduled later. Start time ***T*** of the later task will be greater than or equal to the start time ***T*** plus the processing time of the task scheduled prior. Here, since job 1 is scheduled prior to job 2, the following constraint holds:

$$\boldsymbol{T_{1,3} + 2 \leq T_{2,3}}$$

However, if job 2 was scheduled prior to job 1, then the following constraint holds:

$$\boldsymbol{T_{2,3} + 3 \leq T_{1,3}}$$

Now, let $P_{i,j}$ denote processing time for a task *t* (*i, j*). For the task *t (1, 2)*, which corresponds to pair (2, 2) in job 1, $P_{i,j}$ is 2. The end time for task *t* (*i, j*) is given by $\boldsymbol{T_{i,j} + P_{i,j}}$. Therefore,

the length of a solution to the job shop problem is the maximum of start time summed with the processing time of all the tasks. Given as,

$$\max_{\substack{i = 1 \text{ to } m \\ j = 1 \text{ to } n}} \left\{ T_{i,j} + P_{i,j} \right\}$$

i is a job, m is the total number of jobs ($m \geq 1$)
j is a task; n is the total number of tasks ($n \geq 1$)
$T_{i,j}$ is the start time for j^{th}task of job i, which is a variable
$P_{i,j}$ is the processing time for j^{th}task of job i

$$T_{i,j} = \begin{cases} T_{i,j-1} + P_{i,j-1} & \text{if } i = 1 \\ T_{i-1,j} + P_{i-1,j} & \text{if } i > 1 \end{cases}$$

$T_{1,1}$= 0 [Base case]

CHAPTER V - RESULTS

The Dynamo code extracts total number of elements, Type IDs, Names and Element IDs of all the elements in the selected Revit model. It also calculates the distances between centroids of the elements and creates a list. The Dynamo code developed for extracting this information from 3D Revit models produces the data required by the TSP program in JESS to determine the generalized schedule of optimal path.

5.1 TSP schedule for Test cases

Using the data extracted from Dynamo files, the JESS program incorporating TSP algorithm gave the following schedule my three test cases. Below figure 52 shows the TSP schedule for test case 1.

NW → EW1 → MW1 → WW → SW → MW2 → EW2

1 → 2 → 3 → 4 → 5 → 6 → 7

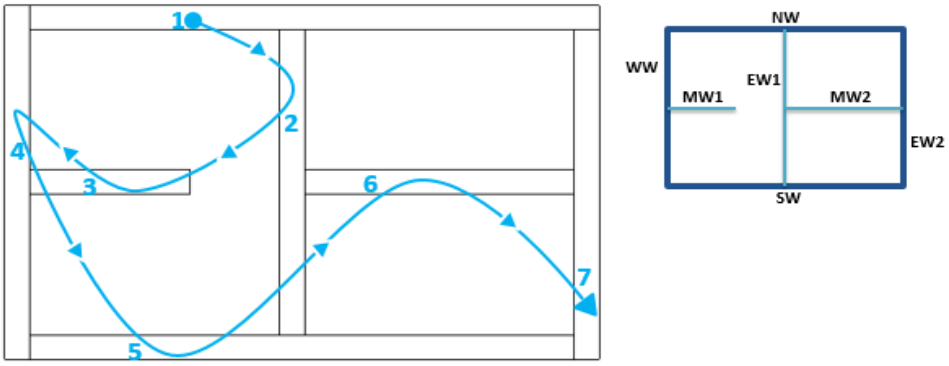

Figure 52 TSP schedule for Test Case 1

Below Table 2 Schedule Recommendation and Costs for Test Case 1 by Human Expert 1, Human Expert 2 and Expert System compares the cost of the schedules provided by two human experts with the schedule produced by the expert system for the test case 1, which contains 7 walls (4 exterior & 3 interior).

Table 2 Schedule Recommendation and Costs for Test Case 1 by Human Expert 1, Human Expert 2 and Expert System

	Wall	Human Expert 1	Human Expert 2	Expert System	
Exterior Walls	SW	1	1	5	
	EW2	2	3	7	
	NW	3	2	1	
	WW	4	4	4	
Interior Walls	EW1	5	5	2	
	MW2	6	6	6	
	MW1	7	7	3	Error
COST		**51.73**	**58.3**	**33.14**	

The expert system tries to build the Mid Wall (MW) before finishing the exterior walls, which is considered as an error.

Below figure 53 shows the TSP schedule for test case 2.

MW2 → EW2 → NW1 → SW1 → WW → MW1 → EW1 → SW2 → NW2 → MW2

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

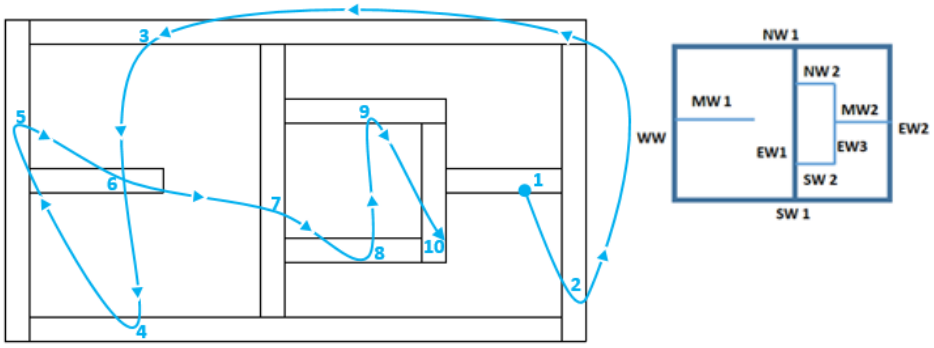

Figure 53 TSP schedule for Test case 2

Below Table 3 compares the costs of the schedules provided by two human experts with the schedule produced by the expert system for the test case 2, which contains 10 walls (4 exterior &6 interior).

Table 3 Schedule Recommendation and Costs for Test Case 2 by Human Expert 1, Human Expert 2 and Expert System

	Wall	Human Expert 1	Human Expert 2	Expert System	
Exterior Walls	SW 1	1	1	4	
	EW2	2	3	2	
	NW1	3	2	3	
	WW	4	4	5	
Interior Walls	EW1	5	5	7	
	SW2	6	6	8	
	NW2	7	7	9	
	EW3	8	8	10	
	MW2	9	9	1	Error
	MW1	10	10	6	
COST		**57.2**	**63.95**	**48.28**	

The expert system started scheduling with the interior wall before finishing the exterior walls, which is considered error.

Figure 54 below shows the TSP schedule for test case 3.

EW4 → WW → NW1 → MW1 → EW1 → SW1 → SW2 → NW2 → EW3 → MW2 → EW2 → SW3

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12

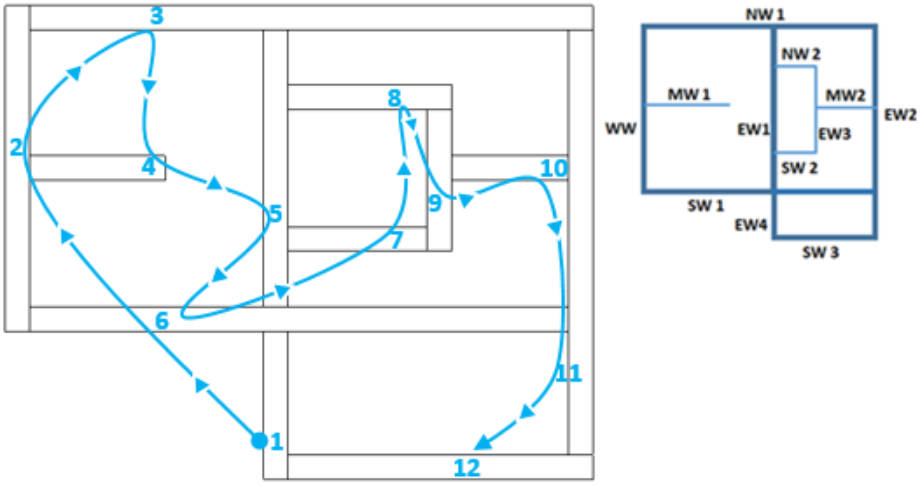

Figure 54 TSP schedule for Test case 3

The below Table 4 compares the costs of the schedules provided by two human experts with the schedule produced by the expert system for the test case 3, which contains 12 walls (6 exterior & 6 interior).

Table 4 Schedule Recommendation and Costs for Test Case 3 by Human Expert 1, Human Expert 2 and Expert System

	Wall	**Human Expert 1**	**Human Expert 2**	**Expert System**	
Exterior Walls	SW 1	4	4	1	
	EW4	5	6	6	
	SW3	6	7	2	
	EW2	3	3	3	
	NW1	1	1	11	Error
	WW	2	2	12	Error
Interior Walls	MW1	12	12	4	Error
	EW1	7	5	5	
	SW2	8	9	7	
	EW3	10	10	8	
	MW2	11	11	9	
	NW2	9	8	10	
COST		**80.88**	**93.19**	**68.27**	

The expert system's recommendation jumps to interior walls before finishing the exterior walls. This is considered as an error in real life construction projects.

5.2 JSP Schedule for Test cases

This generated schedule from JESS, human expert knowledge and machines/processing time information is provided to the Job Shop Problem algorithm developed using Python programming language. The developed python code is executed on an open source software, PyCharm. The generalized schedule of jobs obtained from JESS expert system is saved in the same directory as this Python code in a text (.txt) file format. When the main function is executed, the program asks for the file name/path of file containing the jobs to be scheduled. Once the system extracts the order of jobs, it then

collects the sequence of machines for every individual job, and the processing time for each individual task of all jobs. Once all the required input values are provided, as seen in figure 55, the system computes the optimal schedule for machines to finish all the assigned tasks within the given timeframe. The system also provides the time intervals of all tasks for each machine (figure 56). For test case1, the optimal schedule length is 20.

```
C:\Users\Feroz\AppData\Local\Programs\Python\Python37\python.exe C:/ProjectJSP/code_v6.py
Enter jobs file path: jobs1.txt
Processing for job sequence: 3 4 6 1 2 7 5
Provide machine sequence for processing Job 3 tasks: 1 2 3 5
Enter the processing times for job 3 tasks: 2 1 2 1
Provide machine sequence for processing Job 4 tasks: 1 2 3 4 5
Enter the processing times for job 4 tasks: 2 2 1 1 3
Provide machine sequence for processing Job 6 tasks: 1 3 4 5
Enter the processing times for job 6 tasks: 1 1 2 2
Provide machine sequence for processing Job 1 tasks: 1 2 3 4 5
Enter the processing times for job 1 tasks: 2 3 1 2 1
Provide machine sequence for processing Job 2 tasks: 1 2 4 5
Enter the processing times for job 2 tasks: 1 2 1 2
Provide machine sequence for processing Job 7 tasks: 1 2 3 4
Enter the processing times for job 7 tasks: 3 2 3 2
Provide machine sequence for processing Job 5 tasks: 1 3 5
Enter the processing times for job 5 tasks: 2 2 2
```

Figure 55 Machine sequence and Process times

```
Optimal Schedule

Carpenter (060100): : job_3_1   job_4_1   job_6_1   job_1_1   job_2_1   job_7_1   job_5_1
Insulator (072100): : job_3_2   job_4_2   job_1_2   job_7_2   job_2_2
Electrician (260500): : job_3_3   job_6_2   job_4_3   job_1_3   job_7_3   job_5_2
Finisher (060600): : job_4_4   job_6_3   job_1_4   job_2_3   job_7_4
Assembler (081700): : job_3_4   job_4_5   job_6_4   job_1_5   job_2_4   job_5_3

Time Intervals of Tasks

Carpenter (060100): : [0,2]     [2,4]     [4,5]     [5,7]     [7,8]     [8,11]    [11,13]
Insulator (072100): : [2,3]     [4,6]     [7,10]    [11,13]   [13,15]
Electrician (260500): : [3,5]     [5,6]     [6,7]     [10,11]   [13,16]   [16,18]
Finisher (060600): : [7,8]     [8,10]    [11,13]   [15,16]   [16,18]
Assembler (081700): : [5,6]     [8,11]    [11,13]   [13,14]   [16,18]   [18,20]
Optimal Schedule Length: 20
```

Figure 56 Optimal Schedule and Length of solution

For test case 2, the optimal schedule length is 26 (figure 58).

```
C:\Users\Feroz\AppData\Local\Programs\Python\Python37\python.exe C:/ProjectJSP/code_v6.py
Enter jobs file path: jobs2.txt
Processing for job sequence: 9 2 1 6 3 7 4 5 10 8
Provide machine sequence for processing Job 9 tasks: 1 2 3 4 5
Enter the processing times for job 9 tasks: 2 1 2 1 3
Provide machine sequence for processing Job 2 tasks: 1 2 3 5
Enter the processing times for job 2 tasks: 1 2 3 1
Provide machine sequence for processing Job 1 tasks: 1 2 3 4 5
Enter the processing times for job 1 tasks: 2 1 2 3 1
Provide machine sequence for processing Job 6 tasks: 1 2 4 5
Enter the processing times for job 6 tasks: 3 1 1 3
Provide machine sequence for processing Job 3 tasks: 1 2 4 5
Enter the processing times for job 3 tasks: 2 1 3 1
Provide machine sequence for processing Job 7 tasks: 1 3 4 5
Enter the processing times for job 7 tasks: 1 1 1 2
Provide machine sequence for processing Job 4 tasks: 1 4 5
Enter the processing times for job 4 tasks: 2 2 2
Provide machine sequence for processing Job 5 tasks: 1 2 3 4
Enter the processing times for job 5 tasks: 2 1 2 1
Provide machine sequence for processing Job 10 tasks: 1 2 3 4 5
Enter the processing times for job 10 tasks: 2 3 2 3 1
Provide machine sequence for processing Job 8 tasks: 1 4 5
Enter the processing times for job 8 tasks: 2 2 2
```

Figure 57 Machine Sequence and Process times

```
Optimal Schedule

Carpenter (060100): : job_9_1   job_2_1   job_1_1   job_6_1   job_3_1   job_7_1   job_4_1   job_5_1   job_10_1  job_8_1
Insulator (072100): : job_9_2   job_2_2   job_1_2   job_6_2   job_3_2   job_5_2   job_10_2
Electrician (260500): : job_9_3   job_2_3   job_1_3   job_7_2   job_5_3   job_10_3
Finisher (060600): : job_9_4   job_6_3   job_1_4   job_3_3   job_7_3   job_4_2   job_5_4   job_8_2   job_10_4
Assembler (081700): : job_9_5   job_2_4   job_6_4   job_1_5   job_3_4   job_7_4   job_4_3   job_8_3   job_10_5

Time Intervals of Tasks

Carpenter (060100): : [0,2]     [2,3]     [3,5]     [5,8]     [8,10]    [10,11]   [11,13]   [13,15]   [15,17]   [17,19]
Insulator (072100): : [2,3]     [3,5]     [5,6]     [8,9]     [10,11]   [15,16]   [17,20]
Electrician (260500): : [3,5]     [5,8]     [8,10]    [11,12]   [16,18]   [20,22]
Finisher (060600): : [5,6]     [9,10]    [10,13]   [13,16]   [16,17]   [17,19]   [19,20]   [20,22]   [22,25]
Assembler (081700): : [6,9]     [9,10]    [10,13]   [13,14]   [16,17]   [17,19]   [19,21]   [22,24]   [25,26]
Optimal Schedule Length: 26
```

Figure 58 Optimal Schedule and Length of solution

For test case 3, the optimal schedule length is 30 (figure 60).

```
C:\Users\ferozahmed\AppData\Local\Programs\Python\Python37\python.exe C:/PythonJSP/code_v6.py
Enter jobs file path: jobs.txt
Processing for job sequence: 1 3 6 4 5 2 8 7 9 10 11 12
Provide machine sequence for processing Job 1 tasks: 1 2 3 4
Enter the processing times for job 1 tasks: 3 2 2 2
Provide machine sequence for processing Job 3 tasks: 1 2 3 4
Enter the processing times for job 3 tasks: 2 1 1 2
Provide machine sequence for processing Job 6 tasks: 1 2 3 4
Enter the processing times for job 6 tasks: 3 2 2 2
Provide machine sequence for processing Job 4 tasks: 1 2 5
Enter the processing times for job 4 tasks: 3 2 2
Provide machine sequence for processing Job 5 tasks: 1 2 3 5
Enter the processing times for job 5 tasks: 2 1 1 2
Provide machine sequence for processing Job 2 tasks: 1 2 3 4
Enter the processing times for job 2 tasks: 2 3 3 2
Provide machine sequence for processing Job 8 tasks: 1 2 5
Enter the processing times for job 8 tasks: 1 1 1
Provide machine sequence for processing Job 7 tasks: 1 2 5
Enter the processing times for job 7 tasks: 1 2 1
Provide machine sequence for processing Job 9 tasks: 1 2 4
Enter the processing times for job 9 tasks: 2 1 2
Provide machine sequence for processing Job 10 tasks: 1 3 4
Enter the processing times for job 10 tasks: 1 1 2
Provide machine sequence for processing Job 11 tasks: 1 2 3 4
Enter the processing times for job 11 tasks: 2 1 1 3
Provide machine sequence for processing Job 12 tasks: 1 2 3 4
Enter the processing times for job 12 tasks: 2 2 1 1
```

Figure 59 Machine Sequence and Process times

```
Optimal Schedule

Carpenter (060100): : job_1_1   job_3_1   job_6_1   job_4_1   job_5_1   job_2_1   job_8_1   job_7_1   job_9_1   job_10_1  job_11_1  job_12_1
Insulator (072100): : job_1_2   job_3_2   job_6_2   job_4_2   job_5_2   job_2_2   job_8_2   job_9_2   job_7_2   job_11_2  job_12_2
Electrician (260500): : job_1_3   job_3_3   job_6_3   job_5_3   job_2_3   job_10_2  job_11_3  job_12_3
Finisher (060600): : job_1_4   job_3_4   job_6_4   job_9_3   job_2_4   job_10_3  job_11_4  job_12_4
Assembler (082517): : job_4_3   job_5_4   job_8_3   job_7_3

Time Intervals of Tasks

Carpenter (060100): : [0,3]     [3,5]     [5,8]     [8,11]    [11,13]   [13,15]   [15,16]   [16,17]   [17,19]   [19,20]   [20,22]   [22,24]
Insulator (072100): : [3,5]     [5,6]     [8,10]    [11,13]   [13,14]   [15,18]   [18,19]   [19,20]   [20,22]   [22,23]   [24,26]
Electrician (260500): : [5,7]     [7,8]     [10,12]   [14,15]   [18,21]   [21,22]   [23,24]   [26,27]
Finisher (060600): : [7,9]     [9,11]    [12,14]   [20,22]   [22,24]   [24,26]   [26,29]   [29,30]
Assembler (082517): : [13,15]   [15,17]   [19,20]   [22,23]
Optimal Schedule Length: 30
```

Figure 60 Optimal Schedule and Length of solution

Below, Table 5 shows the comparison of completion times to compute the optimal schedule length by JSP solver and an alternative MILP (Mixed-Integer Linear Programming) solver CPLEX (run on the NEOS server [44]). The MILP model is adopted from [47], which can only produce approximate solutions to our JSP instances, since it assumes that all jobs must pass through all machines in sequence; i.e., even if it can skip a machine, it is assumed to pass through that machine spending 0 time in its queue. However, if the machine happens

to be busy at that time serving another job, then the pass-through job has to wait, yielding a (possibly) worse solution than our JSP solver. Both the MILP model (written in AMPL---a mathematical optimization programming language [49]) and the CPLEX solutions from NEOS are included in the Appendix.

Table 5 Comparison of completions times by JSP solver and CPLEX

Test Cases	**JSP Solver**	**CPLEX**
Test case 1	20	21
Test case 2	26	29
Test case 3	30	31

CHAPTER VI - CONCLUSION AND FUTURE DIRECTIONS

By analyzing the results from test cases, I can conclude that my computational framework consisting of JESS expert system incorporated with TSP algorithm, combined with JSP algorithm can help in automating and expediting the process of construction scheduling. This framework can save a considerable amount of time and resources.

However, there are certain limitations to this framework. One of the limitations is, according to the construction industry human experts, outer/exterior walls must be scheduled prior to interior walls. My expert system schedules interior walls before the exterior walls in certain cases, which is considered an error. One potential approach to solve this limitation might work as follows: first identify the convex hull of the model/wall coordinates, and then identify the models/walls that touch this convex hull. Such walls could be called the exterior walls, and rest of the walls the interior walls. Although this heuristic may not be accurate in certain cases, it might work very well in practice in distinguishing the exterior walls from the interior walls. Once this distinction has been made, my TSP solver can be applied separately to the two types of walls, and the schedule of the exterior walls can precede that of the interior walls. In order to ensure that the distance between the last exterior wall and the first interior wall does not adversely impact the overall solution quality, an additional constraint might be applied to the Hill Climbing algorithm that ensures that the first interior wall for the second run of the TSP solver is the closest neighbor of the last exterior wall returned from the first call of the TSP solver.

Experts in the construction field have identified an alternative optimization objective for the Job Shop problem. This objective seeks to minimize the interference among the machines (e.g., carpenter, electrician, etc.), such that when one machine is active

at a job site (wall), no other machine is allowed to be active within a certain distance. Currently, my framework aims at keeping the total makespan of the jobs to a minimum, the traditional objective for JSP. Due to this, all the machines perform their tasks immediately one after the other without any excessive wait time. But this could cause a problem in real life construction projects. For example, if two different machines (carpenter & electrician) are working on adjacent jobs, there could be interruptions caused by the workspace interference. This could eventually delay the work of machines. And, in real life construction projects, an electrician is called into work on his tasks only after the carpenter finishes all his tasks. This limitation could possibly be solved by adding a constraint in the Job Shop Problem to let one machine finish its task on all the jobs before the next machine starts working on its tasks. This will increase the total makespan of the jobs, but it will address one of the main requirements that is important in the construction field.

Among other limitations, certain input steps, like providing the machine sequence for jobs and processing times for machines to the JSP algorithm, is still partly a manual process. Automating this input process could really advance the functionality of this expert system.

Further research into adding the above-mentioned constraints to the algorithms and automating the input process could make this framework exceptionally helpful to construction engineers by increasing the efficiency of their schedules and reducing the time taken to prepare these schedules.

APPENDIX

Alternate solutions of job shop problem implementation for my test cases gives a lengthier solution. For test cast 1, the length of the solution obtained here is 21, on contrary to my program's solution 20.

```
# SETS

param n;

param m;

set I := 1..n;

set K := 1..m;

# PARAMETERS

param p{i in I, k in K}, >=0;

param d{i in I}, >=0;

param bigM := max{i in I}(d[i]);

param y{i in I, j in I, k in K}, binary;

# VARIABLES

var t{i in I, k in K}, >=0;

#var y{i in I, j in I, k in K}, binary;

var eta, >=0;

# OBJECTIVE FUNCTION

minimize makespan: eta;

# CONSTRAINTS

subject to eta_max{i in I}:

eta >= t[i,m] + p[i,m];
```

subject to deadline{i in I}:

t[i,m] + p[i,m] <= d[i];

subject to disjunction_1{i in I, j in I, k in K: i<j}:

t[i,k] + p[i,k] <= t[j,k] + bigM*(1-y[i,j,k]);

subject to disjunction_2{i in I, j in I, k in K: i<j}:

t[j,k] + p[j,k] <= t[i,k] + bigM*y[i,j,k];

subject to sequence{i in I, k in (K diff {m})}:

t[i,k] + p[i,k] <= t[i,k+1];

Solution:

t[i,k] t[i,k] + p[i,k] =

1 1	5	7
1 2	7	10
1 3	10	11
1 4	11	13
1 5	13	14
2 1	7	8
2 2	10	12
2 3	12	12
2 4	13	14
2 5	14	16
3 1	0	2
3 2	2	3
3 3	3	5
3 4	5	5
3 5	5	6
4 1	2	4
4 2	4	6
4 3	6	7
4 4	7	8
4 5	8	11
5 1	11	13
5 2	14	14
5 3	17	19
5 4	19	19
5 5	19	21

```
6 1     4         5
6 2     6         6
6 3     7         8
6 4     8        10
6 5    11        13
7 1     8        11
7 2    12        14
7 3    14        17
7 4    17        19
7 5    19        19
;

t[i,k] [*,*] (tr)
:     1    2    3    4    5    =
1    5    7   10   11   13
2    7   10   12  13   14
3    0    2    3    5    5
4    2    4    6    7    8
5   11   14  17  19   19
6    4    6    7    8   11
7    8   12   14   17   19
```

For test case 2, the length of the solution obtained here is 29, on contrary to my program's solution 26.

```
t[i,k] t[i,k] + p[i,k] =
1 1     3         5
1 2     5         6
1 3     8        10
1 4    10        13
1 5    13        14
2 1     2         3
2 2     3         5
2 3     5         8
2 4     8         8
2 5     9        10
3 1     8        10
3 2    10        11
3 3    11        11
3 4    14        17
3 5    17        18
4 1    11        13
4 2    13        13
4 3    13        13
4 4    18        20
```

4 5 20 22
5 1 13 15
5 2 15 16
5 3 16 18
5 4 20 21
5 5 22 22
6 1 5 8
6 2 8 9
6 3 10 10
6 4 13 14
6 5 14 17
7 1 10 11
7 2 11 11
7 3 11 12
7 4 17 18
7 5 18 20
8 1 17 19
8 2 20 20
8 3 22 22
8 4 25 27
8 5 27 29
9 1 0 2
9 2 2 3
9 3 3 5
9 4 5 6
9 5 6 9
10 1 15 17
10 2 17 20
10 3 20 22
10 4 22 25
10 5 25 26

t[i,k] [*,*] (tr)

:	1	2	3	4	5	=
1	3	5	8	10	13	
2	2	3	5	8	9	
3	8	10	11	14	17	
4	11	13	13	18	20	
5	13	15	16	20	22	
6	5	8	10	13	14	
7	10	11	11	17	18	
8	17	20	22	25	27	
9	0	2	3	5	6	
10	15	17	20	22	25	

:	t[i,m] + p[i,m]	d[i]	:=
1	14	500	
2	10	500	
3	18	500	
4	22	500	
5	22	500	
6	17	500	
7	20	500	
8	29	500	
9	9	500	
10	26	500	

For test case 3, the length of the solution obtained here is 31, on contrary to my program's solution 30.

t[i,k]	t[i,k] + p[i,k]	=
1 1	0	3
1 2	3	5
1 3	5	7
1 4	7	9
1 5	9	9
2 1	13	15
2 2	15	18
2 3	18	21
2 4	21	23
2 5	23	23
3 1	3	5
3 2	5	6
3 3	7	8
3 4	9	11
3 5	11	11
4 1	8	11
4 2	11	13
4 3	13	13
4 4	14	14
4 5	14	16
5 1	11	13
5 2	13	14
5 3	14	15
5 4	15	15
5 5	16	18
6 1	5	8
6 2	8	10
6 3	10	12
6 4	12	14

6 5 14 14
7 1 16 17
7 2 19 21
7 3 21 21
7 4 23 23
7 5 24 25
8 1 15 16
8 2 18 19
8 3 21 21
8 4 23 23
8 5 23 24
9 1 17 19
9 2 21 22
9 3 22 22
9 4 23 25
9 5 25 25
10 1 19 20
10 2 22 22
10 3 22 23
10 4 25 27
10 5 27 27
11 1 20 22
11 2 22 23
11 3 23 24
11 4 27 30
11 5 30 30
12 1 22 24
12 2 24 26
12 3 26 27
12 4 30 31
12 5 31 31
t[i,k] [*,*] (tr)
: 1 2 3 4 5 =
1 0 3 5 7 9
2 13 15 18 21 23
3 3 5 7 9 11
4 8 11 13 14 14
5 11 13 14 15 16
6 5 8 10 12 14
7 16 19 21 23 24
8 15 18 21 23 23
9 17 21 22 23 25
10 19 22 22 25 27
11 20 22 23 27 30
12 22 24 26 30 31

t[i,m]	+	p[i,m]	d[i]	=
1		9	500	
2		23	500	
3		11	500	
4		16	500	
5		18	500	
6		14	500	
7		25	500	
8		24	500	
9		25	500	
10		27	500	
11		30	500	
12		31	500	

REFERENCES

[1] A. Berrais and A. Watson, "Expert systems for seismic engineering: the state-of-the-art," Engineering Structures, vol. 15, pp. 146-154, 1993.

[2] Abdullahi, M., et al. (2008) "A review on expert systems for concrete mix design," ICCBT. A (21), pp. 231-238.

[3] Ahmed, M, Omotunde, H. (2012). Theories and Strategies of Good Decision Making. International Journal of Scientific and Technology Research. 1. 51-54.

[4] Aho, V., et al. (1988). The AWK Programming Language. Addison-Wesley Publishing Company. ISBN 9780201079814.

[5] Akram, M., Rahman, I. A., Memon, I (2014). A Review on Expert System and its Applications in Civil Engineering. International Journal of Civil Engineering and Built Environment Vol.1, No.1, 2014; ISSN 2289-6317

[6] Anoop Gupta (1984). Implementing OPS5 production systems on DADO. Carnegie Mellon University Research Showcase. Retrieved March 2017 from http://repository.cmu.edu/compsci.

[7] Aini, A., Salehipourb, A. (2012). Speeding up the Floyd–Warshall algorithm for the cycled shortest path problem. Applied Mathematics Letters Volume 25, Issue 1, Pages 1-5.

[8] AUTODESK. (2016). Revit Family | BIM Software | Autodesk. Retrieved October 31, 2016, from http://www.autodesk.com/products/revit-family/overview.

[9] Bejar, J. (2010). CLIPS - Code Snippets Version 0.8 (0.8). California 94305, US: Creative Commons. Retrieved on January 2016 from

http://www.cs.upc.edu/~bejar/ia/material/laboratorio/clips/CLIPS-snippets-eng.pdf

[10] Benjamin, C. O., Babcock, D. L., Yunus, N. B. and Kincaid, J (1990) "Knowledge-based prototype for improving scheduling productivity." J. Comp. in Civ. Engrg., ASCE, Vol. 4, No. 2, pp. 124-134.

[11] Berrais, A and Watson, A. (1993), "Expert systems for seismic engineering: the state-of-the-art" Engineering Structures, vol. 15, pp. 146-154.

[12] Berlioz, C (2011). How Rete Algorithm Works. Retrieved January 2016 from https://www.sparklinglogic.com/rete-algorithm-demystified-part-2/

[13] Brownston, L., et al. (1985). Programming Expert Systems in OPS5. Addison-Wesley Publications ISBN 0-201-10647-7

[14] Bykov, Y. & Petrovic, S. J Sched. (2016) A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling Vol:19, pp: 479-492. https://doi.org/10.1007/s10951-016-0469-x

[15] Chabini, I and Lan, S (2002) Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 3, NO. 1.

[16] Chen DS, Batson RG and Dang Y (2010). Applied Integer Programming: Modeling and Solution. John Wiley & Sons: New York.

[17] Graham, S.M., Joshi, A. & Pizlo, Z. The traveling salesman problem: A hierarchical model (2000) 28: 1191. https://doi.org/10.3758/BF03211820

[18] Cerrone, C., Cerulli, R & Golden, B (2017). Carousel Greedy: A Generalized

Greedy Algorithm with Applications in Optimization. Computers & Operations Research. 85.10.1016/j.cor.2017.03.016.

[19] Corke, Bruce H. (2016) Building Information Modeling (BIM) – Expert Article on Construction Disputes. Retrieved October 31, 2016, from http://www.robsonforensic.com/expertshttp://www.robsonforensic.com/articles/building-information-modeling-BIM-expert-witness

[20] CLIPS Reference Manual. Volume I Basic Programming Guide Version 6.30 (2015). Retrieved February 17, 2018, from http://clipsrules.sourceforge.net/documentation/v630/bpg.pdf

[21] Damme, E. V., Kuhn, H et al. (1996). The Work of John Nash in Game Theory. Journal of economic theory 69, 153-185.

[22] Dasgupta S, Papadimitriou CH and Vazirani UV (2006). Algorithms. McGraw-Hill: New York.

[23] Dijkstra's Algorithm. Retrieved February 2018, from http://mathworld.wolfram.com/DijkstrasAlgorithm.html

[24] Dobson, H. V JR (1990). “Expert Systems: A Primer for the Construction Manager”. University of Florida.

[25] Drools - Business Rules Management System (2006). Retrieved February 2018, from https://www.drools.org/

[26] Durkin, J. (2002). History and applications. In C. T. Leondes, & C. T. Leondes (Ed.), Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century (Vol. I, pp. 1-22). USA: Academic Press. doi:10.1016/B978-012443880-4/50045-4

[27] Egan, Andy. (Jan 2007). Some Counterexamples to Causal Decision Theory. The Philosophical Review, Vol. 116, No. 1, pp. 93-114.

[28] Engelmore, R. S., & Feigenbaum, E. (2017). Expert Systems and Artificial Intelligence. Retrieved April 25, 2017, from http://www.wtec.org/loyola/kb/c1_s1.htm

[24] Faghihi, V., Reinschmidt, K F., Kang, J H (2014) “Construction scheduling using Genetic Algorithm based on Building Information Model. Expert Systems with Applications 41 7565–7578

[25] Fondahl JW (1961). A non-computer approach to the critical path method for the construction industry. Technical Report No. 9, The Construction Institute, Department of Civil Engineering, Stanford University, Stanford, CA.

[26] Forgy, C. L., Artificial Intelligence (1982). "Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem” (17-37)

[27] Friedman-Hill, E. (2003). JESS in Action: Java Rule-Based Systems (In Action series): Ernest Friedman-Hill: 9781930110892: Amazon.com: Books. Manning Publications.

[28] GeeksforGeeks | Bellman Ford Algorithm| Retrieved March 2018 from https://www.geeksforgeeks.org/bellman-ford-algorithm-simple-implementation/

[29] GeeksforGeeks | Hill Climbing | Retrieved January 2017 from https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

[30] Gibbons, R (1996). An Introduction to Applicable Game Theory. The Journal of Economic Perspectives, Vol. 11, No. 1. (Winter, 1997), pp. 127-149.

[31] Guo H, Zhu K, Ding C and Li L (2010). “Intelligent optimization for project scheduling of the first mining face in coal mining.” Expert Systems with Applications 37(2): 1294–1301.

[32] Graphisoft. (2016). Open BIM. Retrieved October 31, 2016, from http://www.graphisoft.com/archicad/open_bim/about_bim/

[33] Han, Sang-Yun and Tschangho John Kim (1989). “Expert Systems in Planning (Urban)" Journal of the American Planning Association, pp. 455-477.

[34] Jaskowski, P., Sobotka, A (2006). “Multicriteria Construction Project Scheduling Method Using Evolutionary Algorithm”. An International Journal. Vol.6, No 3, pp.283-297

[35] Kaetzel, L. J., Clifton, J. R., (1995). "Expert/knowledge-based systems for materials in the construction industry: State-of-the-art report," Materials and Structures, vol. 28, pp. 160-174.

[36] Kelley JE and Walker MR (1959). Critical-path planning and scheduling. In: Proceedings of the Eastern Joint Computer Conference. New York, USA, pp 160–173.

[37] Klansek, U (2011). “Using the TSP Solution for Optimal Route Scheduling in Construction Management” DOI 10.5592/otmcj.2011.1.3 Research paper.

[38] Khatib, L and Blanco, J J., "Course Scheduling as A Constraint Satisfaction Problem" Retrieved in November 2019 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.8460&rep=rep1&type=pdf

[39] L. J. Kaetzel and J. R. Clifton, "Expert/knowledge-based systems for materials in the construction industry: State-of-the-art report," Materials and Structures, vol. 28, pp. 160-174, 1995

[40] Lewis, D (1981). Causal Decision Theory. Australasian Journal of Philosophy Vol. 59, No. 1.

[41] Liao, S-H. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. Expert Systems with Applications 28 (2005) 93–103.

[42] Matai, R., Singh, S P and Mittal M L (2010). Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, Traveling Salesman Problem, Theory and Applications, Prof. Donald Davendra (Ed.), ISBN: 978-953-307-426-9.

[43] Martínez-Rojas María, Marín Nicolás, & Vila M. Amparo. (2016). The Role of Information Technologies to Address Data Handling in Construction Project Management. Journal of Computing in Civil Engineering, 30(4), 04015064. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000538

[44] McGartland, M. R., & Hendrickson, C. T. (1985, September). Expert Systems for Construction Project Monitoring. Journal of Construction Engineering and Management, 111(3), 293-307. doi:10.1061/(ASCE)0733-9364(1985)111:3(293)

[45] Montibeller G., Franco A. (2010) Multi-Criteria Decision Analysis for Strategic Decision Making. In: Zopounidis C., Pardalos P. (eds) Handbook of Multicriteria Analysis. Applied Optimization, vol 103. Springer, Berlin, Heidelberg

[46] Neos Solvers. Retrieved in August 2019 from https://neos-server.org/neos/solvers/index.html

[47] Opricovic, Serafim., Tzeng, Gwo-Hshiung. (2002). Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS, European Journal of Operational Research 156 (2004) 445–455.

[48] Planning, Scheduling in Construction Management. (2015). Retrieved February 21, 2017, from http://theconstructor.org/construction/planning-scheduling-and-construction-management/14/

[49] Production Scheduling. Retrieved August 2019, from http://home.deib.polimi.it/taccari/OPT_14-15/scheduling.pdf

[50] Rao, R V. (2008). A decision-making methodology for material selection using an improved compromise ranking method. Materials & Design - MATER DESIGN. 29. 1949-1954. 10.1016/j.matdes.2008.04.019.

[51] Robert Fourer, David M. Gay and Brian W. Kernighan, “A Modeling Language for Mathematical Programming.” Management Science 36 (1990) 519–554.

[52] Rouse, M (2010). Expert System. Retrieved February 20, 2017, from http://searchhealthit.techtarget.com/definition/expert-system

[53] Saoud, E. A. (1996), Expert Systems for Management Training in the Construction Industry.

[54] Sniedovich, M (2006). Dijkstra’s algorithm revisited:the dynamic programming connexion. Control and Cybernetics vol. 35, No. 3, pp: 599-620.

[55] Sulbaran, T., & Ahmed, F. (2017). Expert System for Construction Scheduling Decision Support Based on Travelling Salesman Problem. 53rd ASC Annual International Conference Proceedings. Retrieved from http://ascpro.ascweb.org/chair/paper/CPRT223002017.pdf

[56] Tang, S., Ahmad, I., Ahmed, S., & Lu, M. (2004). PLANNING AND SCHEDULING DECISIONS. In Quantitative Techniques for Decision Making in Construction (pp. 163-180). Hong Kong University Press. Retrieved from http://www.jstor.org/stable/j.ctt2jc6xz.15

[57] Tang, P., Mukharjee, A., and Onder, N (2013). Construction Schedule Simulation for Improved Project Planning: Activity Criticality Index Assessment. Proceedings of the 2013 Winter Simulation Conference.

[58] Takeshi Yamada, & Ryohei Nakano. (1997). Genetic algorithms in engineering systems - Job-shop scheduling, pp 134–160.

[59] Technopedia | Retrieved June 18, 2019 from https://www.techopedia.com/definition/17137/genetic-algorithm

[60] The Job Shop Problem | Optimization. (2017). Retrieved April 27, 2017, from https://developers.google.com/optimization/scheduling/job_shop

[61] The Rete Algorithm | Retrieved March 14, 2019, from https://www.jessrules.com/docs/71/rete.html

[62] Tyler Riddell. (2017, February 10). The Importance of Scheduling – Why Scheduling Software is Imperative for Successful Projects. Retrieved from https://esub.com/the-importance-of-scheduling-why-scheduling-software-is-imperative-for-successful-projects/

[63] Waugh LM and Froese TM (1991). Constraint knowledge for construction scheduling. First International Conference on Expert Planning Systems, Brighton, UK, pp 114–118.

[64] What is decision making? Definition and meaning - BusinessDictionary.com. (2017). Retrieved April 25, 2017, from http://www.businessdictionary.com/definition/decision-making.html

[65] Weisstein, Eric W (2016). "Traveling Salesman Problem." From MathWorld-A Wolfram Web Resource http://mathworld.wolfram.com/TravelingSalesmanProblem.html

[66] Wideman, R. M. (2004). How to motivate all stakeholders to work together. In D. I. Cleland (Ed.), Field guide to management (2nd ed., pp. 288–304). Hoboken, NJ: Wiley.

[67] Yao, X. & Li, G. J. (1991) General simulated annealing. Journal of Computer Sci. & Technol. 6: 329. https://doi.org/10.1007/BF02948392

[68] Zou, R., Liu, Y., et al. REILP Approach for Uncertainty-Based Decision Making in Civil Engineering. Journal of Computing in Civil Engineering Vol. 24, Issue 4 (July 2010)

[69] Zavichi, A; Madan, K; Xanthopoulos, P and Oloufa, (2013) A “Tsp-Based Model for On-Site Material Handling Operations with Tower Cranes”.

[70] Zwass, V. (2016). Expert system | computer science. Retrieved February 23, 2017, from https://www.britannica.com/technology/expert-system.