Université de Montréal

Interrelated Product Design Activities Sequencing with Efficient Tabu Search Algorithms

par
Vlad Lucian Laza

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

avril, 2016

RÉSUMÉ

Ce mémoire présente un nouvel algorithme métaheuristique de recherche taboue pour trouver des solutions optimales ou sous-optimales au problème de minimisation de la longueur des dépendances d'une matrice de conception (FLMP). Ce problème comporte une fonction économique non-linéaire et il appartient à la classe $\mathscr{NP} - ardu$. Il s'ensuit qu'il est très difficile à trouver une solution optimale exacte en temps réel pour les problèmes de taille moyenne ou grande.

D'abord, on présente le problème et une revue de la littérature associée. Ensuite, on analyse le problème, et on présente les détails du nouvel algorithme de recherche taboue produisant des solutions au problème de réduction de l'effet de retour en utilisant deux voisinages différents, le premier basé sur l'échange des positions de deux activités ("swap"), et le second sur le déplacement d'une activité à une position différente ("shift"). Des résultats numériques permettent d'analyser le comportement de l'algorithme et de comparer les deux voisinages. La première étape consiste à déterminer de bonnes valeurs pour les paramètres en utilisant des problèmes générés aléatoirement. Ensuite nos résultats sont comparés avec ceux obtenus dans la littérature.

On conclut que l'algorithme de recherche taboue proposé est très prometteur, car nos résultats sont meilleurs que ceux publiés dans la litérature. D'autant plus que la recherche taboue semble avoir été utilisée pour la première fois sur ce problème.

Mots clés : gestion de projet, métaheuristique, le probleme de minimisation du longeur des dépendances, matrice de conception.

ABSTRACT

This paper proposes and investigates a metaheuristic tabu search algorithm (TSA) that generates optimal or near optimal solutions sequences for the feedback length minimization problem (FLMP) associated to a design structure matrix (DSM). The FLMP is a non-linear combinatorial optimization problem, belonging to the $\mathscr{NP} - hard$ class, and therefore finding an exact optimal solution is very hard and time consuming, especially on medium and large problem instances.

First, we introduce the subject and provide a review of the related literature and problem definitions. Using the tabu search method (TSM) paradigm, this paper presents a new tabu search algorithm that generates optimal or sub-optimal solutions for the feedback length minimization problem, using two different neighborhoods based on swaps of two activities and shifting an activity to a different position. Furthermore, this paper includes numerical results for analyzing the performance of the proposed TSA and for fixing the proper values of its parameters. Then we compare our results on benchmarked problems with those already published in the literature.

We conclude that the proposed tabu search algorithm is very promising because it outperforms the existing methods, and because no other tabu search method for the FLMP is reported in the literature. The proposed tabu search algorithm applied to the process layer of the multidimensional design structure matrices proves to be a key optimization method for an optimal product development.

Keywords: project management, metaheuristic, feedback length minimization problem, design structure matrix

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DSM	Design Structure Matrix
FLMP	Feedback Length Minimisation Problem
MDM	Multi Domain Design structure Matrix
PD	Product Developpment
TFL	Total Feedback Length
TL	Tabu List
TLD	Tabu List Duration
TS	Tabu Search
TSA	Tabu Seach Algorithm
TSM	Tabu Seach Method
TSASH	Tabu Seach Algorithm Based on "shifts" Neighbor Moves
TSASW	Tabu Seach Algorithm Based on "swaps" Neighbor Moves
UCAV	Unmanned combat aerial vehicle

(For the 50^{th} Anniversary of DIRO) *Ad multos annos*.

ACKNOWLEDGMENTS

First I would like to thank Dr. Jaques Ferland for his continuous support, encouragement and patience that he showed throughout my research. I would also like to thank my research directors from McGill University, mostly Dr. Amina Lamghari and secondly Dr. Roussos G. Dimitrakopoulos, for the great interest they showed in the first year of my research. Also, I thank Celine Begin for her outstanding secretary work, and the libraries of Concordia and UDEM that hosted me during my studies.

Finally, I would like to express my immense gratitude to my father who introduced me to the universe of mathematics at an early age.

CHAPTER 1

INTRODUCTION

In today's highly concurrent markets, efficient product design is crucial to the success of many corporations and businesses [16] [18]. In recent years, many industries and engineering organizations have focused on concurrent product design in order to quickly introduce new products on the market [5], using various optimization methods. The product development should be organized in three dimensions: the product to be developed, the process involved in developing the product and the teams in charge to execute the various aspects of the project [35].

Whenever a project manager has to design a new product, he needs to rearrange the sequence of interrelated design activities of the process in the best possible way, in order to reduce the feedback length between the activities. If the interrelated activities are not arranged in the proper order, the feedback of the process increases, which requires rework and thus decreases the project's performance. Therefore, the following question arises: how to properly sequence a set of interrelated activities in order to minimize the total feedback length of the process activities? The objective of this study is to deal with this question, which is extremely important to design managers.

Concurrent product engineering aims at reducing rework, time and costs of projects [21], [18]. It increases the quality and the reliability of the project products, making the entire project easier to manage [17]. However, an efficient concurrent product design may increase the complexity of the process development, due to interactions between the activities in the process [21].

Advances in operations research and computer science, during the past three decades, have led to many new optimization techniques for dealing with complex product development. The tabu search (TS) [14] can be used to solve a broad range of optimization problems. The approach in this paper consists in applying the TS to the research area of complex product development, where theoretical foundations and significant progress have been accomplished.

The design process starts with identifying the requirements for the project [21]. Once the activities that need to be executed within the process are identified, the information flow dependencies between the activities are estimated and recorded in a design structure matrix (DSM). Furthermore, the DSM is analyzed using a chosen optimization technique and a sequence of reordered interrelated activities is finally produced. The designer has to follow the execution order of all the activities given by this sequence.

The examined problem of DSM sequencing can be formulated as a feedback length minimization problem (FLMP), which is known to be a $\mathscr{NP} - hard$ combinatorial optimization problem [27]. Consequently, the methods that attempt to solve the FLMP in an exact matter quickly become inefficient and time-consuming, as the size of the problem increases [27], [35]. It seems that the tabu search method is not used at all in the literature to deal with the FLMP.

In 2012, Yassine [35] mentions that "tabu search must be considered for medium or large DSM problems" and also that "it is difficult to achieve further improvements within a single domain (DSM)". Qian [27] also mentioned in 2013 the need for using the tabu search in DSM-based sequencing. Thus this paper aims to bring further improvements to the effectiveness of single domain process DSM optimization by extending the application of the tabu search method to this field. We implement the technique and we test its efficiency using real-life and simulated problems.

A review of the literature is presented in the following chapter. The problem formulation is further introduced in Chapter 3. The proposed tabu search algorithm and its variations are described in Chapter 4. By running the algorithm on a large number of different problems, a set of numerical results is obtained and presented in Chapter 5, along with parameter fixing and benchmark analysis.

The tabu search method can generate very good solutions in reasonable amounts of time, even though we cannot guarantee their optimality. In this paper, the proposed tabu search method is shown to outperform other methods found in the literature, to solve the benchmarked problems. In the final chapter, an overview of the main research is discussed, along with indications for future research.

CHAPTER 2

RELATED LITERATURE

Planning a sequence of many activities requires an identification and a resequencing of the interrelated activities [9] [32]. This chapter first introduces the design structure matrix (DSM), which is a powerful analysis tool for system modeling. The DSM is increasingly used in scheduling interrelated activities of projects [27]. A multitude of DSM-based optimization techniques for sequencing interrelated activities were found in the literature. A common objective of some researchers is finding an activity sequence that minimizes the total feedback length (TFL) associated to a DSM matrix. This specific type of problem is known as the feedback length minimization problem (FLMP) [27]. Several researchers focused on solving the FLMP, using other methods than the tabu search, as mentioned in the second part of this chapter. The next chapter about the problem formulation also includes other relevant references.

2.1 The Design Structure Matrix

The DSM is a powerful tool used to model elements comprising a complex system and the interactions between the elements, providing a global image of the system's structure (or architecture). Donald Steward developed the early DSM technique in the 1960s for finding solutions to systems of equations, focusing on the order of the variables to be solved and on minimizing the solution algorithm's iteration. Steward realized that the DSM can be applied to the sequencing of activities in processes, by giving a better representation and improvement of the sequence of activities. His work on the DSM was

published in 1981 [32]. Warfield [34] used a binary matrix representation of systems. Starting in the early 1990s, researchers (several of them from MIT) improved the early DSM models and applied the concepts to industrial problems. The DSM was used by Rogers for the NASA [29], Grose [15] for Boeing, and Black [4] and Eppigner [8] for General Motors. The use of design structure matrices for modeling complex processes has increased since, becoming the largest field of DSM research and application.

The DSM is a N x N square matrix used in product development to model process architecture, where the DSM elements are the interrelated activities involved in the process and the interactions are the information flows between these activities. This study is focused on optimizing complex process DSMs that involve many interrelated sequential activities. The process can be viewed as a system of activities and their interactions required in the development of a project. The activities are all the tasks to be executed in the process. Interactions between the activities are modeled as information flows, where information outputs of some activities influence the execution of other activities. The process architecture DSM is also known as activity-based DSM, process flow DSM or simply process DSM [7]. The sequencing is completed trough an analysis of the DSM and an ordering of the activities which are sequential in our model. Iteration or rework occurs when the activities need to be repeated. Two or more activities are coupled if their interactions create iteration. Coupled activities are also known under the name of cycles or feedback loops and can deteriorate the process performance if they are not executed in the proper order. The DSM matrix has row and colum headings both representing the same index sequence of interrelated activities. The diagonal elements are empty and divide the DSM into two triangular portions. The subdiagonal elements of the DSM matrix represent forward information flows from upstream activities to downstream activities denoted as feedforwards, and the superdiagonal entries represent backward information flows from downstream activities to upstream activities, denoted as feedbacks [27]. The

IR/FAD DSM convention specifies that the feedback marks of the DSM are above its diagonal (FAD) and the inputs of the DSM are in rows (IR) and the outputs are in columns [7]. A general representation of the DSM is found in Figure 2.1.

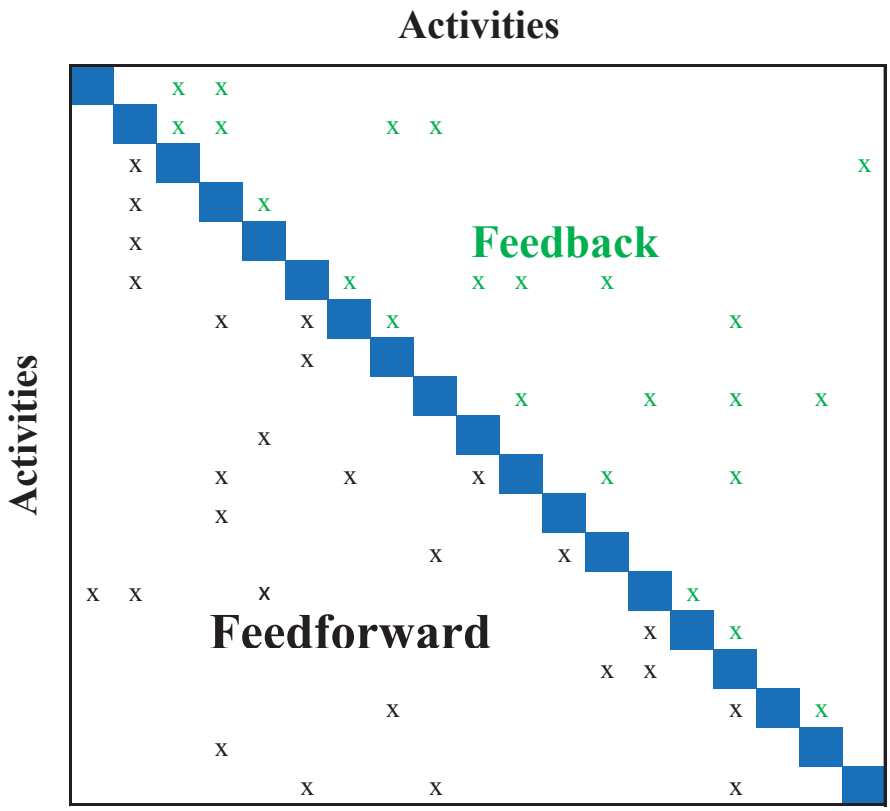

Figure 2.1 – DSM respecting the IR/FAD convention

The length of the feedbacks ("distance between the corresponding downstream and upstream activities") can cause rework in a project, and it can substantially increase the process execution time and costs [27]. For example, a feedback from an activity located at the end of the process activity sequence to an activity located at the beginning of the activity process sequence causes more rework than if the two activities in feedback were located next to each other (neighbors) in the sequence.

Eppigner [7] further classifies the DSM problem models in three categories : static

architectures that represent system elements that exist simultaneously, such as products and organization; temporal flow models, representing system elements that can change trough time, and multidomain matrices that are a combination of multiple DSMs of the previous two (single-domain) models.

Multidomain DSM matrices recently became extremely important because they capture different aspects of the project development (PD) and are required to organize complex projects that take into account products, people and processes [35]. We will see in the next chapters that the proposed tabu search method can be applied to optimize single-domain process DSMs, which can often be components of the multidomain matrix.

2.2 Interrelated Activity Scheduling Techniques

Project management scheduling techniques such as "Program Evaluation and Review Technique (PERT)" [19] and "Critical Path Method (CPM)" [24] emerged during the 50s, and they were used by Western industrial and military organizations to schedule and control complex projects, such as chemicals plants and submarine weapon systems. Their model is based on network flow representation. Even if these techniques were very handful at that time, they cannot deal with information flow cycles [20], [31], which are very common in the context of projects where the activities are interrelated. If the execution of an activity A influences the execution of activity B and if the execution of B also influences the execution of A, then activities A and B are in an information flow cycle or circuit. Researchers such as Smith and Eppigner [30], Krishnan and Ulrich [20], revealed that these early techniques are inefficient.

Many researchers focused on the problem of scheduling interrelated activities. Kusiak and Wang [21] proposed a graph theory based heuristic. The problem of scheduling

interrelated activities was also examined by Abdelsalam and Bao [1] using Simulated Annealing for finding the optimized activity sequence, by Qian and Lin [28] using a hybrid algorithm for large problem sizes and also by Lin [23] with a "fuzzy approach" to schedule interrelated activities with uncertain dependencies. Gebala and Eppinger [11] and other researchers mentioned that, in addition to the number of feedbacks in the DSM, the feedback length should also be considered when sequencing interrelated activities. Qian and Lin [27] refer to this problem as feedback length minimization problem (FLMP). Meyer [25] *et. al* used Genetic Algorithms to order the sequence of the activities for the FLMP formulated as a quadratic assignment problem. Lancaster and Cheng [22] used an Adaptive Evolutionary Algorithm in order to bring the DSM matrix into a lower triangular form and minimize the total feedback length (TFL). Ahmadi *et. al* [2] proposed a hybrid heuristic for finding an activity sequence with minimum total feedback length. Todd [33] used a Genetic Algorithm for minimizing the TFL and he tested his algorithm on several literature problems. Rogers [29] incorporated a Genetic Algorithm into the NASA's DSM tool DeMAID (Design Manager's Aid for Intelligent Decomposition). DeMAID was reducing the iteration (TFL) by reordering the DSM to get as close as possible to a lower triangular form. Qian and Lin [27] proposed a heuristic for finding good initial sequences for the FLMP that can be used as an initial solution, to be improved by the CPLEX MILP-solver, when the size of the problem is small enough (up to 12 activities).

CHAPTER 3

PROBLEM FORMULATION

In this chapter, the DSM matrix is described in more detail and multiple formulations of the feedback length minimization problem are presented. Furthemore, Qians' formulation [27] is reduced to an equivalent, more simple and adequate formulation to adapt the tabu search procedure in Chapter 4.

3.1 Details of the Design Structure Matrix

The information dependencies among interrelated activities can be represented by an activity-activity incidence matrix (DSM) where each element of the matrix represents the relationship between the corresponding pair of activities.

Assume that a design project requires to sequence M activities and among them, N activities are interrelated $(M \geq N)$. The $M-N$ unrelated activities can be excluded from the model because they can be placed at any position in the optimal activity sequence index. In other words, the $M-N$ activities may be executed at any time during the project, as they do not account in the calculation of the TFL.

Let $a_{i,j}$ be the degree of information dependency of activity i on activity j where $0 \leq a_{i,j} \leq 1$, $i \leq N$ and $j \leq N$. Note that the elements located on the diagonal of the DSM matrix are left blank because by definition, an activity information cannot depend on itself. Some researchers use the elements located on the diagonal to denote the activity index [25]. Each element of the DSM represents an activity-activity dependence

relationship. A general representation of the DSM is given in Table 3.1

Activities	**1**	**...**	***i***	**...**	***j***	**...**	***N***
1		...	$a_{1,i}$	...	$a_{1,j}$	...	$a_{1,N}$
...	...		...	...	...	...	...
i	$a_{i,1}$	...		...	$a_{i,j}$	...	$a_{i,N}$
...	...	...	...		...	...	...
j	$a_{j,1}$	...	$a_{j,i}$	...		...	$a_{j,N}$
...	...	...	...	...	...		...
N	$a_{N,1}$	...	$a_{N,i}$	...	$a_{N,j}$	...	

Table 3.1 – DSM general representation

In Steward [32], the DSM represents only strict precedence relationships. Therefore, the information dependencies values are limited to the values 0 and 1, generating a binary DSM. But in many project development (PD) problems, the dependencies among the interrelated activities are not necessarily of equal weight: interrelated activities can have degrees of information dependency that are not necessarily 0 and 1, but are real numbers in the interval $[0,1]$. Real number DSM elements offer a better modeling of the FLMP because the dependencies can have different strengths.

In practice, the term $a_{i,j}$ is defined as the product of the variability of task j and the sensitivity of task i to j [1],[30]. In information theory, the term $a_{i,j}$ is quantified as the volatility of a task, denoted by Vol and defined as: $Vol = IV * TS$, where IV is the information variability ("variability of the input of information needed") and TS is the task sensitivity ("receiving task's sensitivity to change in that information") [7].

3.2 Formulation of the Feedback Length Minimization Problem (FLMP)

The objective function of the FLMP is to minimize the TFL of the DSM associated to a sequence of activities. There exist several formulations of the FLMP in the literature, each having its own specific objective function and constraints [35]. In this paper, we use the formulation presented by Qian and Lin [27]. It will be shown that their formulation can be reduced to another one, found in Gebala and Eppigner [11] and also in Todd [33].

3.2.1 Qian's Formulation

For a better understanding of the model presented by Qian and Lin [27], we summarize the notation used in this model:

N : the number of interrelated activities in the project ;

A : matrix of information dependencies ;

$a_{i,j}$: the degree of information dependency of activity i on activity j ($0 \leq a_{i,j} \leq 1$) ;

x : solution specifying the precedence among the activities ;

x_{ij} : the binary variables such that $x_{ij} = \begin{cases} 1, & \text{if activity } i \text{ precedes activity } j \text{ ;} \\ 0, & \text{otherwise.} \end{cases}$.

$C(x)$: the objective function evaluating the TFL.

The model is summarized in the following page.

$$\text{(P1)} \quad \text{Minimize} \quad C(x) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{i,j} x_{ij} \left(\sum_{\substack{k=1 \\ k \neq j}}^{N} x_{kj} - \sum_{\substack{k=1 \\ k \neq i}}^{N} x_{ki} \right)$$

Subject to

$$x_{ij} + x_{ji} = 1, \quad \text{for } 1 \leq i < j \leq N \tag{1}$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2, \quad \text{for all distinct activities } i, j \text{ and } k \tag{2}$$

$$x_{ij} \in \{0,1\} \quad \text{for } 1 \leq i, 1 \leq j, i \neq j \tag{3}$$

The objective function is to minimize the total feedback length, $C(x)$, specified as a quadratic function, by approaching the interrelated activties in the solution sequence. If activity i precedes activity j, the term $x_{ij} \left(\sum_{\substack{k=1 \\ k \neq j}}^{N} x_{kj} - \sum_{\substack{k=1 \\ k \neq i}}^{N} x_{ki} \right)$, denoting the length of a feedback is greater than or equal to 0. If activity i does not precede activity j, then $x_{ij} = 0$ and the term does not contribute to the objective value. Constraint (1) states that if activity i precedes activity j, then activity j cannot precede activity i and vice versa. Constraint (2) ensures the transitivity property : if activity i precedes activity j and activity j precedes activity k, then activity i must precede activity k. Constraint (3) defines x_{ij} as binary numbers.

3.2.2 A More Simple Formulation of the FLMP

The sequence of interrelated activities can be represented as an array $X = (X[1], ..., X[N])$, where the k^{th} entry of X (denoted by $X[k]$) is the index of the activity located at the position k of the array X. Note that the array X is a permutation of $\{1, 2, ..., N\}$. Hence, $X[k] \in \{1, 2, ..., N\}$. The permutation is representing an ordering of the activities. To establish the equivalence between the two objective functions, we refer to the position

of activity i denoted by $p_i \in \{1,2,...,N\}$. Using this new notation, the problem $(P1)$ reduces to the following problem $(P2)$:

$$\text{(P2)} \quad \text{Minimize} \quad C = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} a_{X[i],X[j]}\,(j-i)$$

over all the permutations X of $\{1,2,...,N\}$. To verify the equivalence, suppose that x_{ij}, $1 \leq i,j \leq N$, $i \neq j$ is a solution of problem $(P1)$. Consider two activities i and j, $i \neq j$. We have

$$a_{i,j}\,x_{ij}\left(\sum_{\substack{k=1\\k\neq j}}^{N} x_{kj} - \sum_{\substack{k=1\\k\neq i}}^{N} x_{ki}\right) = \begin{cases} a_{i,j}\left(\sum_{\substack{k=1\\k\neq j}}^{N} x_{kj} - \sum_{\substack{k=1\\k\neq i}}^{N} x_{ki}\right) & \text{if } x_{ij}=1; \\ 0 & \text{otherwise.} \end{cases}$$

But $x_{ij} = 1$ means that activity i precedes activity j in the corresponding solution formulation $(P1)$. Thus in terms of positions of the activities

$$a_{i,j}\,x_{ij}\left(\sum_{\substack{k=1\\k\neq j}}^{N} x_{kj} - \sum_{\substack{k=1\\k\neq i}}^{N} x_{ki}\right) = \begin{cases} a_{i,j}(p_j - p_i) & \text{if } p_j > p_i\,; \\ 0, & \text{otherwise} \end{cases}$$

and it follows that only the terms i,j where $p_j > p_i$ can have a positive value in the objective function of formulation $(P1)$. Since the objective function in formulation $(P2)$ is specified in terms of the position of the activities in the array X, it follows that :

$$\sum_{i=1}^{N}\sum_{\substack{j=1\\j\neq i}}^{N} a_{i,j}\,x_{ij}\left(\sum_{\substack{k=1\\k\neq j}}^{N} x_{kj} - \sum_{\substack{k=1\\k\neq i}}^{N} x_{ki}\right) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} a_{X[i],X[j]}\,(j-i)$$

This proposed formulation will be used in the rest of this paper due to its simplicity. It is more elegant than the formulation of Qian and Lin since the constraint requires only that the solution be a permutation of $\{1,2,...,N\}$.

3.3 A Practical Example

To illustrate these concepts, we take as example a process DSM used by the Boeing Company to create the conceptual design phase of a UCAV (unmanned combat aerial vehicle), for the US military [7]. The original DSM consists of $N = 12$ interrelated activities and each activity index (from 1 to 12) corresponds to an activity name, given in Table 3.2. For example activity index 2 corresponds to the activity name "Create Configurations Options". The original process DSM is given in Figue 3.1 and feedbacks are shown above its diagonal (see highlighted elements). Using the formulation in Section 3.2.2, we calculate the total feedback length associated with the permutation of $X = \{1,2,3,...,12\}$:

$$C = \sum_{i=1}^{11} \sum_{j=i+1}^{12} a_{X[i],X[j]} \, (j-i) = 34.$$

The initial total feedback length of the DSM can be decreased by rearranging the activities using the tabu search algorithm presented in Chapter 4. The resulting sequence is $\overline{X} = \{1,2,3,8,5,7,4,6,9,11,10,12\}$ and the total feedback length becomes 24. The rearranged DSM associated with $\overline{X}$ is illustrated in Figure 3.2. Because the TFL of the original DSM was reduced to 24, the reordered DSM in Figure 3.2 shows a less iterative and faster process if the activities are executed sequentially in the following order : $1,2,3,8,5,7,4,6,9,11,10,12$.

Activity index	Activity name
1	Prepare UCAV Conceptual DRO
2	Create Configuration Concepts
3	Prepare 3-View Drawing and Geometry Data
4	Perform Aerodynamics Analyses and Eval.
5	Perform Propulsion Analyses and Eval.
6	Perform SC Characteristics Analyses and Eval.
7	Perform Mechanical and Electrical Analyses and Eval.
8	Perfrom Weights Analyses and Eval.
9	Perfrom Perfromance Analyses and Eval.
10	Perform Multidisciplinary Analyses and Eval.
11	Make Concept Assessment and Variant Decisions
12	Prepare and Distribute Choice Config. Data Set

Table 3.2 – Conceptual design of UCAV : activity index and corresponding names

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1		1	0	0	0	0	0	0	0	0	1	0
2	1		0	0	0	0	0	0	0	0	1	0
3	1	1		0	0	0	0	1	0	0	0	0
4	1	1	1		1	0	0	1	0	0	0	0
5	1	1	1	0		0	1	0	0	0	0	0
6	1	0	1	1	1		0	1	0	0	0	0
7	1	0	0	1	1	0		0	0	0	0	0
8	1	0	1	0	0	0	1		0	0	0	0
9	1	0	0	1	1	0	0	1		0	0	0
10	1	1	0	1	1	1	1	1	1		0	0
11	1	1	0	1	1	1	1	1	1	1		0
12	1	1	0	0	0	0	0	1	0	0	1	

Figure 3.1 – Process DSM example (before sequencing)

	1	2	3	8	5	7	4	6	9	11	10	12
1		1	0	0	0	0	0	0	0	1	0	0
2	1		0	0	0	0	0	0	0	1	0	0
3	1	1		1	0	0	0	0	0	0	0	0
8	1	0	1		0	1	0	0	0	0	0	0
5	1	1	1	0		1	0	0	0	0	0	0
7	1	0	0	0	1		1	0	0	0	0	0
4	1	1	1	1	1	0		0	0	0	0	0
6	1	0	1	1	1	0	1		0	0	0	0
9	1	0	0	1	1	0	1	0		0	0	0
11	1	1	0	1	1	1	1	1	1		1	0
10	1	1	0	1	1	1	1	1	1	0		0
12	1	1	0	1	0	0	0	0	0	1	0	

Figure 3.2 – Process DSM example (after sequencing)

CHAPTER 4

TABU SEARCH : METHOD AND PROPOSED ALGORITHM

4.1 Introduction

Assume that we want to schedule a sequence of N distinct interrelated activities. There are $N!$ possible permutations of the activities which correspond to the total number of feasible solutions. Let ζ denote the solution space of the FLMP problem. Notice that $|\zeta| = N!$ and ζ increases factorially as the size N of the problem increases. Experimental results show that the FLMP is extremely hard to solve even for small problem instances. In fact, it is known that the FLMP belongs to the $\mathscr{NP} - hard$ class of problems [27], [35]. Each sequence of activities among the $N!$ possible sequences is feasible, because the interrelated activities can be rearranged in any order. Only one or a few of these permutations will lead to an optimal solution. Therefore, exact optimization algorithms require an exponential time to solve the FLMP, and their inefficiency increases with the size of the problem.

For this reason, we introduce in this chapter a metaheuristic algorithm for solving the FLMP in a reasonable amount of time, even for large instances. We focus on finding a "good" solution sequence and its associated TFL being an upper bound on its optimal value. Thus we evaluate the costs of feasible solution sequences being as close as possible to the optimal solution. Recall from Chapter 3 that we reduced $(P1)$ to $(P2)$ in order to reach an objective function which is easier to evaluate.

4.2 The Tabu Search Method for Solving FLMP

The tabu search is a heuristic method, initially introduced by Glover [13] in 1986 and it can be used for solving complex optimization problems. In Ferland and Costa, [10], several heuristic search methods are summarized, including the tabu search. A general template for simple tabu search as well as implementation guidelines are presented by Gendreau and Potvin [12]. A good detailed presentation of the tabu search is also found in Glover and Laguna [14]. In this solution approach, a local iterative search technique explores the neighborhood of a current solution, looking for a better solution, until a stopping criterion is reached. Each problem has its own specific neighborhood that includes solutions obtained with a slight modification to the current solution. To be more specific, suppose that x is the current solution. Let M be the set of all possible modifications to the current solution. Thus a new solution is obtained using a modification $m \in M$ to generate $x' = x \oplus m$, $m \in M$. So the neighborhood $N(x)$ of the current solution includes all the solutions obtained by using all the different $|M|$ modifications :

$$N(x) = \{x' = x \oplus m \mid m \in M\}$$

In our implementations we will derive two variants of the tabu search using two different neighborhoods, and they will be compared numerically. Recall that in the descent method, the best solution in $N(x)$ is selected to be the new current solution until it is not possible anymore to improve the current solution. Thus, this method stops when it reaches a local minimum of the objective function. In the tabu search procedure, it is allowed to move out of local minima by moving to the best solution in the neighborhood $N(x)$, even if the value of the objective function does not improve. But then we have to prevent cycling (returning to a solution sequence already visited). For this reason, some

modifications are prohibited (become tabu) for a while. Note that the tabu modifications can be used again after a fixed number of iterations because the solutions have changed enough so that these modifications should not generate solutions already visited. Nevertheless, tabu modifications might generate new solutions that were never visited before with a value better than the best solution found so far. In this case, an aspiration criterion allows these modifications. In fact, a simple aspiration criterion often used is the following : allow a tabu modification if it results in a better solution than the best solution found so far.

In general, the tabu search procedure stops using the following two criteria : after a fixed number of iterations or after a fixed number of consecutive iterations without improvement to the objective function.

The procedure starts with an initial solution x^0. In our case, the initial solution is a random permutation of $\{1,2,...,N\}$ obtained by using the Durstenfeld's algorithm [6].

4.3 Mechanics of the Proposed Tabu Seach Algorithm

The general tabu seach procedure is described in Algorithm 1. We introduce two variants of this procedure using different sets of modifications but similar tabu lists. The first one, named **TSASW**, is based on swaps where two activities swap their positions in the permutation. In the second one, named **TSASH**, an activity is shifted to another position in the permutation.

Algorithm 1 The general tabu search procedure

procedure TABU SEARCH

Select an initial random permutation of X'
Let $T \leftarrow \emptyset$ (Tabu list initially empty)
Let $iter \leftarrow 0$; $niter \leftarrow 0$
$X \leftarrow X^0$; $X^* \leftarrow X^0$; $Stop \leftarrow false$
while not Stop **do**
$iter \leftarrow iter + 1$; $niter \leftarrow niter + 1$
Determine a subset $V \subset N(X)$ of solutions $X' = X \oplus m$ satisfying at least one of the two following conditions :
• $m \notin T$
• $C(X') < C(X^*)$
Determine $X'' \in V$ such that $X'' \leftarrow arg\ min_{X' \in V} C(X')$
$X \leftarrow X''$
if $C(X) < C(X^*)$ **then** $X^* \leftarrow X$ and $niter \leftarrow 0$
if $iter = itermax$ or $niter = nitermax$ **then** $Stop \leftarrow true$
Update tabu list

X^* is the best solution generated

4.4 Tabu Search Algorithm based on Swaps (TSASW)

In this variant, a neighbor of a current solution X is obtained by swapping the positions of two activities. More specifically, suppose that the activities in position i and j are swapped :

$X = [\,X[\,1\,],\ldots,X[\,i-1\,],X[\,i\,],X[\,i+1\,],\ \ldots,X[\,j-1\,],X[\,j\,],X[\,j+1\,],\ \ldots,X[\,N\,]\,].$

The corresponding neighbor is :

$X' = [\,X[\,1\,],\ \ldots,X[\,i-1\,],X\,(j\,),X[\,i+1\,],\ldots,X[\,j-1\,],X[\,i\,],X[\,j+1\,],\ldots,X[\,N\,]\,].$

In [27] it is shown that :

$$C(X)-C(X') = \theta_{X[\,i\,],\,X[\,j\,]} = \sum_{l=i+1}^{j-1}[\,(l-i)(a_{X[\,i\,]X[\,l\,]}-a_{X[\,j\,]X[\,l\,]})+(j-l)(a_{X[\,l\,]X[\,j\,]}-a_{X[\,l\,]X[\,i\,]})\,]+(j-i)\,[\,a_{X[\,i\,]X[\,j\,]}-a_{X[\,j\,]X[\,i\,]}+\sum_{l=1}^{i-1}a_{X[\,l\,]X[\,j\,]}-a_{X[\,l\,]X[\,i\,]}+\sum_{l=j+1}^{N}a_{X[\,i\,]X[\,l\,]}-a_{X[\,j\,]X[\,l\,]}\,].$$

The value of the objective function decreases when $\theta_{X[\,i\,],X[\,j\,]} > 0$. Four elements are involved in a swap : activity $X[\,i\,]$, position i in X, activity $X[\,j\,]$, position j in X. Thus, the tabu list must account for these four elements to avoid cycling. It should make sure that activities $X[\,i\,]$ and $X[\,j\,]$ do not return to positions i and j, respectively, for the next TLD iterations.

The tabu list (TL) can be implemented using a square matrix with N (number of activities) rows and N (number of positions in the permutation) columns. Hence, when the swap of $X[\,i\,]$ and $X[\,j\,]$ is completed, the i^{th} element of row $X[\,i\,]$ and the j^{th} element of row $X[\,j\,]$ take both the value (TLD+$iter$) indicating that $X[\,i\,]$ cannot return to position i before iteration (TLD+$iter$) is completed, and similarly for $X[\,j\,]$ to position j.

At some iteration *iter*, the swap $X[\,k\,], X[\,l\,]$ is prohibited (or tabu) if either $TL_{X[\,k\,],l} \geq iter$ or $TL_{X[\,l\,],k} \geq iter$. This strategy implies that activity $X[\,k\,]$ or $X[\,l\,]$ cannot be moved to a position where it was assigned in the last TLD iterations. For a better understanding of the tabu list, assume that the current solution sequence is $X = [2,4,1,3,5]$ and activities 4 and 1 are swapped. The positions of the activities in the array X are $i = 2$ and $j = 3$ and the elements in the tabu list become $TL_{4,2} = iter$+TLD and $TL_{1,3} = iter$+TLD. At the next iteration, the current solution becomes $X = [2,1,4,3,5]$. Assume that at iteration $iter+1$ we want to swap the activities 1 and 4 in the current solution. The algorithm will not allow this swap because $TL_{4,2} = iter$+TLD$> iter+1$ and $TL_{1,3} = iter$+TLD$>$ $iter+1$. The **TSASW** is summarized in Algorithm 2.

4.5 Tabu Search Algorithm based on Shifting (TSASH)

In this variant, a neighbor of a current solution X is obtained by moving an acitvity $X[\,i\,]$ currently in position i to another position $j \neq i$. If $X[\,i\,]$ is moved backward (i.e $i > j$) then

$X = [\,X[\,1\,], X[\,2\,], \ldots, X[\,j-1\,], X[\,j\,], \ldots, X[\,i-1\,], X[\,i\,], X[\,i+1\,] \ldots, X[\,N\,]\,]$

and the neighbor X' is :

$X' = [\,X[\,1\,], X[\,2\,], \ldots, X[\,j-1\,], X[\,i\,], X[\,j\,], \ldots, X[\,i-1\,], X[\,i+1\,], \ldots, X[\,N\,]\,]$

Thus

$X'[\,k\,] = X[\,k\,], k = 1, \ldots, j-1, i+1, \ldots, N$

$X'[\,j\,] = X[\,i\,]$

$X'[\,j+1\,] = X[\,j\,]$

...

$X'[\,i\,] = X[\,i-1\,]$

Algorithm 2 Tabu search algorithm base on swaps (**TSASW**)

procedure TSASW

Select an initial random permutation of X^0
Let $\theta_{i,j} \leftarrow 0$ for $i, j = 1, 2, ..., N$
Let $iter \leftarrow 0$;
$X \leftarrow X^0$; $X^* \leftarrow X^0$; $Stop \leftarrow false$
while not Stop **do**
$iter \leftarrow iter + 1$;
$X'' \leftarrow \emptyset$; $C(X'') \leftarrow +\infty$;$(k_i, kk_i) \neq (k_j, kk_j) \leftarrow (0,0)$
for $i = 1$ to $N-1$ **do**
for $j = i+1$ to N **do**
X' obtained by swapping $X[i]$ and $X[j]$
Compute $\theta_{X[i]X[j]}$
$C(X') = C(X) - \theta_{X[i]X[j]}$
if $C(X') < C(X^*)$ or ($TL_{X[i]j} < iter$ and $TL_{X[j]i} < iter$) **then**
if $C(X') < C(X'')$ **then** $X'' \leftarrow X'$;
$(k_i, kk_i) \leftarrow (X[i], i)$
$(k_j, kk_j) \leftarrow (X[j], j)$
if $X'' = \emptyset$ **then** $Stop \leftarrow true$
else:
$X \leftarrow X''$
if $C(X'') < C(X^*)$ **then** $X^* \leftarrow X''$
if $iter = itermax$ **then** $Stop \leftarrow true$
$TL_{k_i, kk_i} = iter +$ TLD
$TL_{k_j, kk_j} = iter +$ TLD

X^* is the best solution generated

On the other hand, if $X[\,i\,]$ is moved forward (i.e $i < j$) then :

$X = [\,X[\,1\,],X[\,2\,],\,...\,,X[\,i-1\,],X[\,i\,],X[\,i+1\,],\,...\,,X[\,j-1\,],X[\,j\,],X[\,j+1\,]\,...\,,X[\,N\,]\,]$

and the neighbor X'' is :

$X'' = [\,X[\,1\,],X[\,2],\,...\,,X[\,i-1\,],X[\,i+1\,],\,...\,,X[\,j-1\,],X[\,j\,],X[\,i\,],X[\,j+1\,],...,X[\,N\,]\,].$

Thus :

$X''[\,k\,] = X[\,k\,], k = 1,...,i-1,j+1,...,N$

$X''[\,i\,] = X[\,i+1\,]$

$X''[\,i+1\,] = X[\,i+2\,]$

$X''[\,j-2\,] = X[\,j-1\,]$

$X''[\,j-1\,] = X[\,j\,]$

...

$X''[\,j\,] = X[\,i\,].$

As illustrated above, the positions of several activities are modified when we are shifting an activity $X[\,i\,]$ from its current position i to another position j. But to simplify the tabu status, in this case, we prohibit $X[\,i\,]$ to return to the position i during the next the next TLD iterations and $X[\,j\,]$ to return to the position j during the next TLD iterations.

As in the other variant, the tabu list (TL) can be implemented using a square matrix with N (number of activities rows) and N (number of position in the permutation) columns. Since the modification involve shifting an activity $X[\,i\,]$ from its current position i to j, one way to modify the tabu list would be to modify $TL_{X[i]i}$ to the value TLD+$iter$ to prevent $X[\,i\,]$ to return to position i during the next TLD iterations. But preliminary testing indicates that cycling is not prevented in some cases. Indeed, suppose that $X[\,i\,]$ is moved to position $(i+1)$. Then $X[\,i+1\,]$ is moved to position i. Nothing prevents $X[\,i+1\,]$ (currently in position i) to be moved back to position $(i+1)$ at the next iteration. Thus, $X[\,i\,]$ is moved back to position i, and we have a cycle. Note that these

two shifts correspond to a swap. For this reason at each iteration we modify the tabu list TL as in the preceding variant. Hence, to summarize if the modification indicate to shift $X[\,i\,]$ from its current position i to position j, then $TL_{X[i]i} = TL_{X[j]j}$ =TLD+$iter$. The verification to see if a shift is tabu is completed as in the previous variant.

The **TSASH** can be summarized in Algorithm 3.

4.5.1 Diversification Strategies

It is well-known that diversification strategies should be combined with metaheuristic methods like tabu search. These strategies allow to search more extensively the feasible domain of the problem by generating new initial solutions to restart the tabu search.

The diversification is applied in two cases. First, when all the possible moves have the tabu status, the tabu search stops. The second type of diversification occurs when the maximum number of iterations ($iter$) reaches a certain threshold value that will be determined in the following chapter.

In order to allow exploring other portions of the search space, a restart diversification is applied in the two variants. We generate a new random permutation of the current solution using Durstenfields' algorithm [6] to initiate a new application of the TS. All the previous values in the tabu list are erased and the algorithm restarts with an empty tabu list. The current number of iterations, $iter$ is reset to 0. The total number of diversifications to be applied is fixed by the user.

Algorithm 3 Tabu search algorithm based on shifting (**TSASH**)

procedure TSASH

Select an initial random permutation of X^0
Let $TL_{i,j} \leftarrow 0$ for $i, j = 1, 2, ..., N$
Let $iter \leftarrow 0$
$X \leftarrow X^0$; $X^* \leftarrow X^0$; $Stop \leftarrow false$
while not Stop **do**
  $iter \leftarrow iter + 1$
  $X'' \leftarrow \emptyset$; $C(X'') \leftarrow +\infty$;$(k_i, kk_i) \neq (0,0)$
  for $i = 1$ to N **do**
    for $j = 1$ to $N, j \neq i$ **do**
      X' obtained by shifting $X[i]$ to j
      Compute $C(X')$
      if $C(X') < C(X^*)$ or ($TL_{X[i]j} < iter$ and $TL_{X[j]i} < iter$) **then**
        if $C(X') < C(X'')$ **then** $X'' \leftarrow X'$;
          $(k_i, kk_i) \leftarrow (X[i], i)$
          $(k_j, kk_j) \leftarrow (X[j], j)$
  if $X'' = \emptyset$ **then** END
  $X \leftarrow X''$
  if $C(X'') < C(X^*)$ **then** $X^* \leftarrow X''$
  if $iter = itermax$ **then** $Stop \leftarrow true$
  $TL_{k_i, kk_i} = iter$+TLD
  $TL_{k_j, kk_j} = iter$+TLD

X^* is the best solution generated

CHAPTER 5

NUMERICAL RESULTS

5.1 Problem Presentation and Testing Environment

The tabu search algorithm (TSA) performance is greatly influenced by the value of its parameters. Proper calibration of the parameters is essential, but it can be difficult, especially on problems with a large number of activities, due to the time required to validate each parameter and their interactions. A large number of DSMs are used to test the effectiveness of the tabu search algorithm and calibrate its parameters. After, the TSA is compared against problems found in the literature, for example the Austins' DSM [3] [22] [27] [33] consisting of 51 interrelated activities.

The two variants of the tabu search algorithm coded and tested in the C++ environment are compiled using the C++ GNU compiler. Computational experiments were performed on a Acer Aspire T3-710 PC with an Intel Core i7-6700 CPU, under a Windows 10 x64 operating system. Solution spaces can become very large: for example a DSM matrix of 120 activities has a solution space of 6.6895e+198 feasible solution sequences, so efficient memory allocation and adequate data structures influence the computing performance, especially for medium and large instances. For medium (12 or more activities) and large problem size, exact optimal solutions cannot be obtained using the CPLEX solver.

The first part of this section includes numerical results to analyze and to compare the two variants of the tabu search (**TSASW** and **TSASH**) using a large number of randomly

generated problems. In the second part, we use four known benchmark problems to evaluate the efficiency of the **TSASW** and **TSASH** when compared with methods known in the literature.

5.2 Tests completed using Randomly Generated Problems

5.2.1 Part 1 - Tests without Diversification

A set of binary DSMs (each element is either 0 or 1) of different size and density were generated. The files are organized in 5 folders including problems of size $8, 16, 30, 60, 120$. Each folder is composed of 6 files, containing different random DSM problems of different densities $(0.1, 0.2, 0.3,\ 0.4, 0.5, 0.6)$. The DSM problem set and the tested number of instances can be summarized in Table 5.1.

Problem Size	Densities	Number of tested problems of each density
8	0.1, 0.2, 0.3, 0.4, 0.5, 0.6	100
16	0.1, 0.2, 0.3, 0.4, 0.5, 0.6	100
30	0.1, 0.2, 0.3, 0.4, 0.5, 0.6	100
60	0.1, 0.2, 0.3, 0.4, 0.5, 0.6	100
120	0.1, 0.2, 0.3, 0.4, 0.5, 0.6	5

Table 5.1 – DSM problem set

In the literature, practical DSMs are often of density less than 0.5 [27]. The density is the number of nonzero DSM elements divided by N^2, where N is the problem size. Qian and Lin [27] use a similar definition of density: the number of nonzero DSM elements divided by $N(N-1)$. In order to generate a random DSM of a chosen density level, we determine the number of elements different from 0 as the ceiling value of $\lceil density \cdot N^2 \rceil$. Then we randomly generate that number of pairs (x, y) made of two different pseudo-

random integers using the Mersenne Twister pseudo-random generator [26], from the *mt19937* class of the C++ Standard Library based on two pseudo-random seeds. The entries (x, y) of the matrix take a value of 1.

An example of a randomly generated binary DSM matrix of density 0.2, consisting of 8 activities is found in Table 5.2. The initial activity index sequence is by default $[1, 2, 3, 4, 5, 6, 7, 8]$.

	1	0	0	0	1	0	0
0		0	0	1	0	0	1
0	1		0	0	0	1	0
0	0	0		0	0	0	0
1	0	0	1		0	0	0
0	0	0	0	0		1	0
0	1	0	1	0	0		0
1	0	0	0	1	0	0	

Table 5.2 – Example of a data binary DSM

During preliminary tests, a large initial stopping criterion was established for each problem set as presented in Table 5.3. Recall that in our tests, we use only one stopping criterion based on the number of iterations (*itermax*).

	Stopping Criterion	
Problem Size	**TSASW**	**TSASH**
8	200	200
16	400	400
30	800	800
60	1600	1600
120	3200	3200

Table 5.3 – *itermax* value (number of iterations)

First, the values of the parameter tabu list duration, previously denoted by TLD, are examined. To show the effect of the values of this parameter on solution quality, the values of $itermax$ are fixed, for each tested problem size, to the values of the initial stopping criterion in Table 5.3. We solve each problem once with different values of TLD. We select the values of TLD that give best average solutions (minimum TFL) for each problem subset. For simplicity, if there are more than one value of TLD that produces the best average TFL, the smallest value is selected. The values of TLD that minimize the average TFL for each density and problem size are presented in Table 5.4.

		Tabu list duration (number of iterations)				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	5	20	35	110	200
	TSASH	10	65	90	180	200
0.2	**TSASW**	5	15	45	130	200
	TSASH	5	50	95	160	200
0.3	**TSASW**	10	15	30	130	140
	TSASH	10	40	100	170	160
0.4	**TSASW**	5	15	35	110	180
	TSASH	5	35	90	190	200
0.5	**TSASW**	5	15	45	170	180
	TSASH	5	35	80	130	200
0.6	**TSASW**	10	20	35	150	200
	TSASH	10	50	95	190	200

Table 5.4 – Best tabu list durations (TLD)

The TSA is tested on random problems of each density $0.1, 0.2, 0.3, 0.4, 0.5, 0.6$: 100 problems of size 8, 16 and 30 by incrementing the values of TLD from 0 to 100 by a factor of 5; 100 problems of size 60, by incrementing TLD from 0 to 200 by a factor of 10; 5 problems of size 120, by incrementing TLD from 0 to 200 by a factor of 20.

Note that we limit the number of observations of TLD due to the solution times required to solve each problem. For each tested problem subset, the corresponding average TFL can be found in Appendix A, for both **TSASW** and **TSASH**. The results reveal that the tabu list durations TLD increase with the problem size.

The next experimentation was to fix the value of *itermax* to the values of Table 5.3 and the value of TLD to those in Table 5.4, in order to determine the number of iterations until we reach the best solution. Then for each problem size and density, the largest of these numbers over all the problems solved is found in Table 5.5.

		Maximum nb. of iterations till best solution				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	7	354	795	1165	3200
	TSASH	51	325	776	1473	2735
0.2	**TSASW**	32	347	799	1251	3189
	TSASH	23	374	769	1463	2722
0.3	**TSASW**	121	399	799	1554	3200
	TSASH	68	321	769	1507	2003
0.4	**TSASW**	14	340	785	680	2271
	TSASH	18	366	760	1466	2392
0.5	**TSASW**	33	270	790	1228	3041
	TSASH	27	226	771	1284	2780
0.6	**TSASW**	57	285	760	1591	2988
	TSASH	37	233	736	1146	2561

Table 5.5 – Largest number of iterations to reach the best solution

For example, the **TSASW** and the **TSASH** complete at most 32 and 23 iterations, respectively, before reaching the best solution for the 100 tested problems of size 8 and density 0.2. The results in Table 5.5 indicate that this number increases with the problem

size. Referring to the results of this experimentation, we can derive the results in Table 5.6 showing the average (instead of the largest) number of iterations before reaching the best solution. Note that the average number of iterations to reach the best solution is much smaller than the corresponding largest number in Table 5.5. Indeed, this follows from the fact that the largest value is attained only by one or a few problems of the tested problem sets.

		Average nb. of iterations till best solution				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	3	42	243	510	2147
	TSASH	7	23	180	494	1617
0.2	**TSASW**	6	47	245	638	1675
	TSASH	5	40	158	576	1151
0.3	**TSASW**	12	56	225	662	2348
	TSASH	7	35	135	645	922
0.4	**TSASW**	6	45	257	345	1403
	TSASH	6	37	132	457	846
0.5	**TSASW**	6	38	165	413	1542
	TSASH	5	30	140	561	2089
0.6	**TSASW**	7	40	225	558	1717
	TSASH	5	29	140	518	1087

Table 5.6 – Average number of iterations to reach the best solution

Referring to Table 5.5, let us determine the largest value for each size over all the densities. These values are summarized in Table 5.7, and they can be seen as a kind of stopping values for $itermax$. In fact, we observe that they are quite close to the values selected in Table 5.3, with the exception of problems of size 8.

Finally, the elements of Table 5.8 correspond to the average total feedback length for all problem sizes and densities when using the values of Table 5.3 to fix $itermax$,

and those of Table 5.4 for TLD. Note that for any problem size, the TFL increases with the density, and that similarly, for any density, the total feedback length (TFL) increases with the size.

	Stopping Criterion	
Problem Size	**TSASW**	**TSASH**
8	121	68
16	399	374
30	799	776
60	1591	1507
120	3200	2780

Table 5.7 – Observed stopping criteria (number of iterations)

		Average TFL				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	0.86	7.34	109.78	1570.7	16589.2
	TSASH	0.86	7.30	108.57	1567.9	16586.0
0.2	**TSASW**	3.64	48.63	439.82	4480.1	41264.8
	TSASH	3.64	48.46	437.44	4474.1	41271.8
0.3	**TSASW**	11.34	103.22	815.93	7659.7	67883.2
	TSASH	11.34	103.01	815.07	7651.8	67719.4
0.4	**TSASW**	18.70	165.14	1238.5	10939.7	95607.6
	TSASH	18.70	165.04	1236.8	10933.6	95558.4
0.5	**TSASW**	28.20	234.61	1686.8	14481.2	124047.2
	TSASH	28.20	234.54	1684.9	14472.6	124117.2
0.6	**TSASW**	38.26	314.80	2154.4	18357.9	153559.8
	TSASH	38.26	314.72	2153.5	18373.5	153601.2

Table 5.8 – Average total feedback length

We use the two-tailed matched-pairs signed-rank Wilcoxon test to verify if the average TFL produced by the two variants of the tabu search (**TSASW** and **TSASH**)

are different or not. Let $\mu_{\mathbf{TSASH}}$ be the average TFL produced by the **TSASH** and $\mu_{\mathbf{TSASW}}$ be the average TFL produced by the **TSASW**. The null hypothesis is: H_0 : $\mu_{\mathbf{TSASW}} = \mu_{\mathbf{TSASH}}$ and the alternative hypothesis is $H_A : \mu_{\mathbf{TSASW}} \neq \mu_{\mathbf{TSASH}}$. There are 30 pairs of observations, each pair corresponding to the average TFL obtained using the **TSASW** and the **TSASH** respectively, among which 6 pairs are tied (for $N = 8$). The pairs can be seen in Table 5.8. For example, for size 30 and density 0.1 the pair is $(109.78, 108.57)$. The obtained $W - value$ of the Wilcoxon test is 61 and the critical value of W is 81 using a 5% level of confidence. Because the W-value obtained is smaller than the critical value, the null hypothesis is verified, and we conclude that the average TFL obtained using the **TSASW** and the **TSASH** variants of the tabu search is not different. It follows that both variants have the same performance in terms of optimizing the TFL.

		***Avg CPU time* (sec)**				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	0.0009	0.0347	0.9939	32.6097	1033.7751
	TSASH	0.0012	0.0451	1.4218	47.2450	1517.5158
0.2	**TSASW**	0.0012	0.0344	1.0037	32.5409	1028.7433
	TSASH	0.0015	0.0423	1.4059	47.0523	1520.1981
0.3	**TSASW**	0.0010	0.0308	1.0013	32.5859	1026.2587
	TSASH	0.0012	0.0454	1.4265	47.0931	1521.1291
0.4	**TSASW**	0.0012	0.0358	0.9671	28.8066	1029.1321
	TSASH	0.0015	0.0454	1.3573	41.6892	1520.9832
0.5	**TSASW**	0.0011	0.0356	0.9541	28.7909	1028.0662
	TSASH	0.0014	0.0454	1.36	41.3376	1521.4995
0.6	**TSASW**	0.0010	0.0338	0.9093	28.6592	1025.6302
	TSASH	0.0012	0.0431	1.2954	41.2681	1518.6842

Table 5.9 – Average computation times

Another measure of efficiency of the TSA is the average solution time required by

the tabu search, denoted by *Avg CPU*. For each problem size and density, the *Avg CPU* for all the tested problem subsets, is presented in Table 5.9. The average computation times tend to increase as the problem size increases. The Wilcoxon test is not necessary to compare the average solution times of the two variants : we can clearly see that the *Avg CPU* of the **TSASH** is greater than the *Avg CPU* of the **TSASW** for all combinations of problem size and density. The difference in the average computation time of the two variants can be explained by the fact that the **TSASH** has two nested *For* loops that iterate over a larger number of elements than the two nested *For* loops of the **TSASW**.

5.2.2 Part 2 - Tests with Diversification

The diversification previously presented in Section 4.5.1 of Chapter 4 is now applied to analyze the improvement over the total feedback length. The stopping criterion *itermax* is fixed for each problem size to the corresponding value in Table 5.7, each time we apply the procedure. Basically, this selection of *itermax* is to allow the tabu search to perform enough iterations to reach good solutions for all the given densities of a fixed problem size. By restarting the algorithm when reaching this threshold value, we allow exploring other parts of the search space. The tests with diversification are performed using the best values of TLD presented in Table 5.4. We allow a number of 5 diversifications for each tested problem and the results are reported in Table 5.10, for the average total feedback length obtained with the **TSASW** and **TSASH** with diversification.

Next we compare the average total feedback length obtained when using diversification with that in Table 5.8 where diversification is not used. As expected, the average total feedback length (TFL) obtained with diversification is smaller than the TFL computed without diversification, except for $N = 8$ and two instances with $N = 16$, where the problem size is too small to really observe the effects of the diversification. Specifically,

Table 5.11 indicates the percentage of improvement when diversification is applied for each problem size and density.

		Improved Average TFL				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	0.86	7.30	108.14	1565.67	16507.0
	TSASH	0.86	7.30	108.17	1561.56	16510.8
0.2	**TSASW**	3.64	48.45	437.36	4442.54	41262.4
	TSASH	3.64	48.45	436.58	4435.30	41228.6
0.3	**TSASW**	11.34	102.99	814.41	7592.99	67666.6
	TSASH	11.34	102.99	814.25	7585.38	67655.6
0.4	**TSASW**	18.70	165.03	1236.47	10902.7	95528.4
	TSASH	18.70	165.04	1236.62	10893.6	95515.0
0.5	**TSASW**	28.20	234.54	1684.27	14418.2	123912.4
	TSASH	28.20	234.54	1684.21	14412.4	123897.8
0.6	**TSASW**	38.26	314.70	2152.86	18284.1	153460.0
	TSASH	38.26	314.70	2152.84	18279.4	153394.8

Table 5.10 – TFL with diversification

		Percentage of amelioration				
Density	**Tabu Search Algorithm**	$N = 8$	$N = 16$	$N = 30$	$N = 60$	$N = 120$
0.1	**TSASW**	0.000%	0.545%	1.494%	0.320%	0.496%
	TSASH	0.000%	0.000%	0.368%	0.404%	0.453%
0.2	**TSASW**	0.000%	0.370%	0.559%	0.838%	0.006%
	TSASH	0.000%	0.021%	0.197%	0.867%	0.105%
0.3	**TSASW**	0.000%	0.223%	0.186%	0.871%	0.319%
	TSASH	0.000%	0.019%	0.101%	0.868%	0.094%
0.4	**TSASW**	0.000%	0.067%	0.166%	0.338%	0.083%
	TSASH	0.000%	0.000%	0.018%	0.366%	0.045%
0.5	**TSASW**	0.000%	0.030%	0.150%	0.435%	0.109%
	TSASH	0.000%	0.000%	0.043%	0.416%	0.177%
0.6	**TSASW**	0.000%	0.032%	0.073%	0.402%	0.065%
	TSASH	0.000%	0.006%	0.028%	0.512%	0.134%

Table 5.11 – Percentage of improvement of the TFL using diversification

The TSA performs a number of 5 diversifications, so the total number of iterations of the TSA increases by a factor of 6. Observe that the diversification threshold values from Table 5.7 are very close to the initial stopping criterion from Table 5.3 (except for $N = 8$), so the solution time with diversification is around 5 times longer than the solution times presented in Table 5.9, obtained without diversification.

5.3 Solving the Benchmark Problems

Afterward, we solved the benchmarked problems, and we compare our results with those found in the literature. More precisely, we consider the problems of Kusiak (1991) [21] with 12 activities, Steward (1981) [32] with 20 activities, Austin (1996) [22][27] with 51 activities and Qian and Lin (2014) [27] with 48 activities. Austin (1996) problem represents a building design process [3]. Note that a second version of Austin (1996) can be found in [33].

Different authors offered solutions to each of these problems, using methodologies other than tabu search. Kusiak applied his triangulation algorithm to his problem to obtain a solution of 7 (in terms of TFL). Steward obtained a solution for his problem of 93 (in terms of TFL) using a simple algorithm. Tood [33] first solved these benchmark problems in 1997 using his "single criterion Genetic Algorithm with enhanced edge and local search". His algorithm is based on a maximization of concurrency and a minimization of the TFL (also called "iteration"). The maximization of concurrency was done by moving as many of the feedback marks as close as possible to the left hand side or the bottom edge of the matrix. In 2008, Lancaster and Cheng [22] developed the procedure FDAPCEA (Fitness Differential Adaptive Parameter Controlled Evolutionary Algorithm) for a design structure matrix, and they used a mutation operator to generate

diversity in their algorithm. They also validated their algorithm using the three benchmark problems. Qian and Lin [27] proposed in 2014 an exchange-based heuristic to find good initial activity solution sequences for small problem sizes (up to 12 activities) that are then provided to the MILP-solver of CPLEX 12.1. They also found good solutions for problems of large size. A summary of the best results in terms of the TFL obtained by these researchers using different methods for the first three benchmark problems is given in Table 5.12.

Problem	**Size**	**Total Feedback Length**				
		Original Solution	Todd's Solution	Lancaster and Cheng's Solution	Quian's Solution	Optimal Solution
Kusiak (1991)	12	7	6	6	6	6
Steward (1981)	20	93	24	24	24	24
Austin (1996)	51	320	158	157	146	unknown

Table 5.12 – TFL computed using different methods and DSMs

The optimal value of the total feedback length (TFL) is known for the problems of Kusiak (1991) and Steward (1981) as indicated in the last column of Table 5.12. Todd [33], Lancaster and Cheng [22], Qian and Lin [27] provide optimal solutions for the Kusiak (1991) and Steward (1981) problems, and they also provide the rearranged matrices according to their optimal solution sequences. For the Austin (1996) DSM, the optimal value of the TFL is unknown but Qian and Lin provide the best known solution up to now for this problem with a TFL of 146. Our method will be compared with these methods later on in this section.

The most challenging problem is Austin (1996). The initial matrix was not provided by Austin [3]. Unfortunately, two different formulations of the Austin (1996) problem exist in the literature, with a different initial total feedback length. The first one is used

by Lancaster [22] and Qian and Lin [27]. They both provide a reordered Austin (1996) DSM matrix. Lancaster finds a solution sequence with a TFL equal to 157 and Qian and Lin find a solution sequence with a TFL equal to 146 for this problem. They do not provide the initial matrix used for obtaining their solutions. Thus we reordered both of their optimized DSMs to find the one where the activities are ordered from 1 to 51; this ordering has a TFL equal to 300. Furthermore, Todd [33] uses an initial Austins (1996) DSM in his thesis, claiming that the TFL of this matrix is 320. The objective function for computing the TFL used by Todd is similar to the objective function presented in Section 3.2.2. After analyzing this matrix, using Todds' objective function and the objective function presented in Section 3.2.2, we found that the Austins' DSM provided by Todd [33] has an initial TFL of 317, different from the TFL calculated by Todd for this DSM in his thesis, which is 320.

We analyze the problems of Kusiak (1991), Steward (1981), Qian and Lin (2014) and the two versions of Austin (1996) using the two variants of the tabu search and we compare our results (in terms of optimized TFL) with the solutions reported in the literature. We found that Kusiak (1991) DSM has an initial TFL of 39 when the activities are ordered $\{1,2,...,N\}$ and its density is 0.917. Using Table 5.7 as guidance for a problem size of 12, the stopping criterion of the **TSASW** was fixed to 260 iterations $((399+121)/2)$ and the one of **TSASH** to 221 $((68+374)/2)$ iterations by interpolating between the values of known stopping criteria for $N = 8$ and $N = 16$. The known optimal solution of TFL equal to 6 is reached by both variants of the tabu search algorithm, with all the tested values of the tabu list duration (from 0 to 100, incremented by a factor of 5). Our solution sequence and the rearranged DSM of Kusiak (1991) are presented is Table 5.13. The computation times are presented in Table 5.14 and the **TSASH** takes more time to solve the Kusiak (1991) problem than the **TSASW**, although both variants solve this problem very fast.

	1	2	3	11	7	6	10	12	9	8	5	4
1		0	0	0	0	0	0	0	0	0	0	0
2	0		0	0	0	0	0	0	0	0	0	0
3	0	1		0	0	0	0	0	0	0	0	0
11	0	1	1		0	0	0	0	0	0	0	0
7	0	1	0	1		0	0	0	0	0	0	0
6	0	1	0	0	0		0	1	0	0	0	0
10	0	1	1	1	0	1		1	0	0	0	0
12	1	0	0	1	0	0	1		1	0	0	0
9	0	0	1	0	0	1	1	0		0	0	0
8	1	0	0	1	0	0	0	0	1		0	1
5	0	0	0	1	0	1	0	0	0	1		0
4	0	0	1	0	0	1	0	1	0	0	1	

Table 5.13 – Reordered DSM of Kusiak (1991) using tabu search (TLF = 6)

Algoritm	*CPU time (sec.)*		
	Best	Average	Worst
TSASW	0.0011	0.0033	0.0061
TSASH	0.0021	0.0052	0.0067

Table 5.14 – Computation times used by the TSA for Kusiak (1991) DSM

The effects of the diversification on the Kusiak (1991) problem are not interesting since the optimal solution is attained by all the tested values of the tabu list duration.

We found that the Steward (1981) problem when activities are ordered $\{1,2,...,N\}$ has an initial total feedback length (TFL) of 159 and a density level of 0.116. Using Table 5.7 as a guidance for a problem size of 20, and interpolating between the known stopping criteria for $N = 16$ and $N = 30$, the stopping criterion of the tabu search algorithm was fixed to 514 iterations ($\lceil 399 + 4 \cdot (799 - 399)/14 \rceil$) for the **TSASW** and to 518 ($\lceil 374 + 4 \cdot (776 - 374)/14 \rceil$) iterations for the **TSASH**. The tabu list duration was varied from 0 to 100, incremented by a factor of 5. The improvement graph can be found in Figure 5.1.

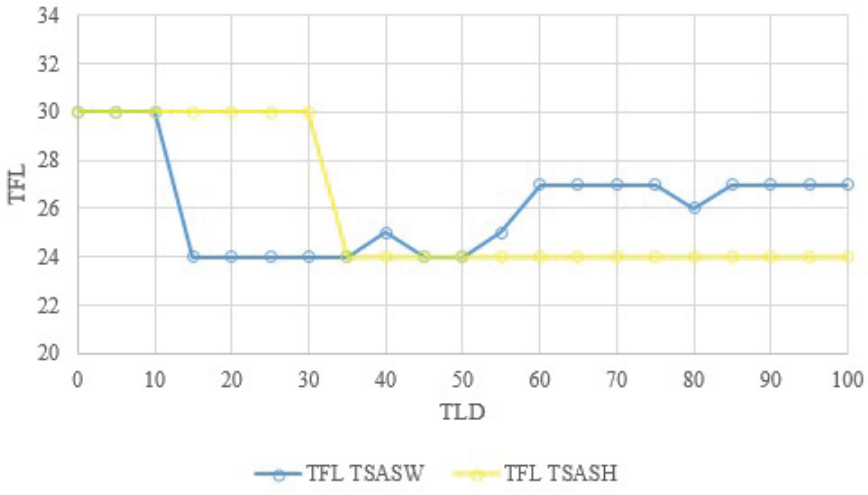

Figure 5.1 – Algorithm improvement over succesive TLD without diversification (Steward 1981)

Figure 5.1 shows that TLD of 15, 20, 25, 30, 35 for the **TSASW** and 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85 ,90, 95 and 100 for the **TSASH** yield optimal solutions, and that the other tested values of the TLD are very close to the optimal solution of 24. Both variants of the tabu search algorithm yield multiple different solution sequences with a TFL of 24. One of the obtained solution sequences and the corresponding rearranged DSM matrix of Steward (1981) is presented in Table 5.16.

The computation times used by the two variants of the TSA for the Steward (1981) problem are presented in Table 5.15.

Algoritm	*CPU time (sec,)*		
	Best	Average	Worst
TSASW	0.0819	0.0873	0.0987
TSASH	0.1044	0.1161	0.1354

Table 5.15 – Computation times used by the TSA procedures for Steward (1981) DSM

	2	**19**	**5**	**6**	**16**	**7**	**8**	**18**	**11**	**9**	**17**	**10**	**4**	**3**	**1**	**15**	**13**	**20**	**14**	**12**
2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	0	1	0		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
7	0	1	1	0	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1		0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	1	0	1	1		0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	0	0	0	0
9	0	1	0	1	0	0	0	1	0		0	1	0	0	0	0	0	0	0	0
17	0	0	0	1	1	0	0	0	0	0		1	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	1	1	0	1		0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1		0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	1	1	0	1		0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0
13	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1		0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
12	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1	

Table 5.16 – Reordered DSM of Steward (1981) using tabu search (TFL = 24)

It will be interesting to observe the effects of the diversification on Stewards' (1981) DSM by varying the tabu list durations from 0 to 100, incremented by a factor of 5. The results obtained with the previously presented diversification are summarized in Figure 5.2. We can see that the diversification improves near optimal solutions obtained without diversification for both **TSASW** and **TSASH**. For almost all the values of the tested TLD, we obtain optimal solutions, except for a TLD equal to 85 with **TSASW**.

Further, we apply the **TSASW** and the **TSASH** to the first formulation of the initial Austin (1996) DSM, with an initial total feedback length (TFL) of 300. A large stopping criterion of 1354 ($\lceil 799 + 21 \cdot (1591 - 799)/30 \rceil$) for the **TSASW** and of 1288

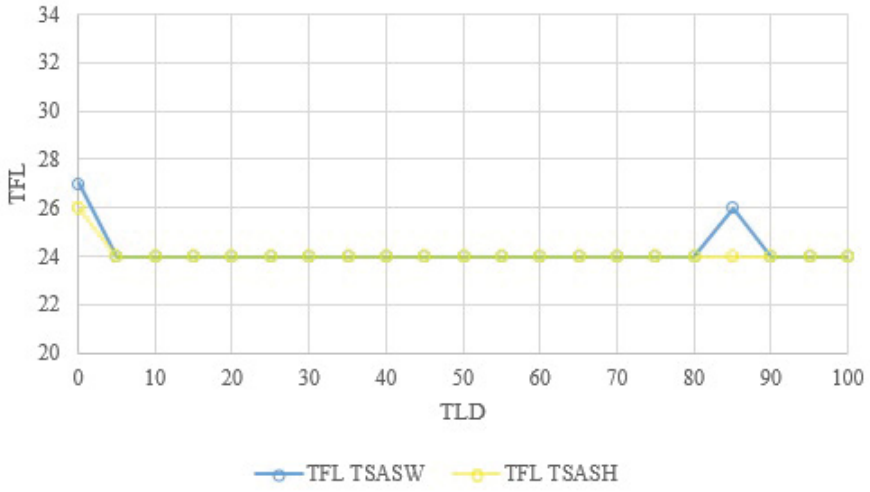

Figure 5.2 – Algorithm improvement over simulated TLD (Steward 1981) with diversification

$(\lceil 776+21\cdot(1507-776)/30\rceil)$ for the **TSASH**, were obtained by interpolating between the stopping criteria values for $N = 30$ and $N = 60$ in Table 5.7 to obtain a value for $N = 51$. The tabu list duration was varied from 0 to 100, incremented by a factor of 5.

The improvement graph of the **TSASW** and **TSASH** can be found is Figure 5.3, and we can see that both **TSASW** and **TSASH** find a best solution sequence with a TFL equal to 133, which is better than all solutions found to date for this problem. The improvement curve shows that tabu list durations of 35, 40, 45 and 60 for the **TSASW** and 45, 50, 55, 60, 65, 70, 75, 80, 85 ,90, 95 and 100 for the **TSASH** yield the best solutions and also that the other tested values of the TLD are very close to our best solution of 133. The reordered DSM matrix of Austin (1996) is found in Table 5.18. The TSA was also applied using diversification to Austins' DSM, and the results are presented in Figure 5.4. We observe that the diversification improves some solutions, for example a solution obtained with a TFL of 135 without diversification was improved to the best solution of 133 for the **TSASW**.

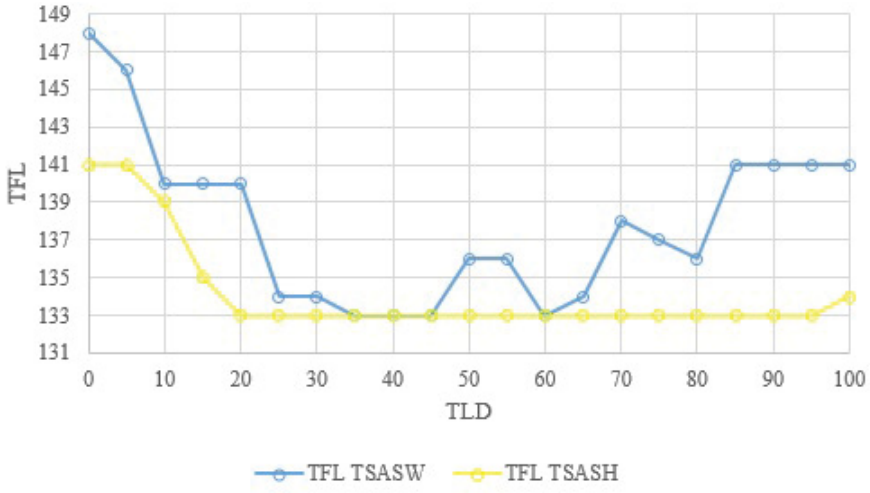

Figure 5.3 – Algorithm improvement over simulated TLD (first formulation of Austin 1996)

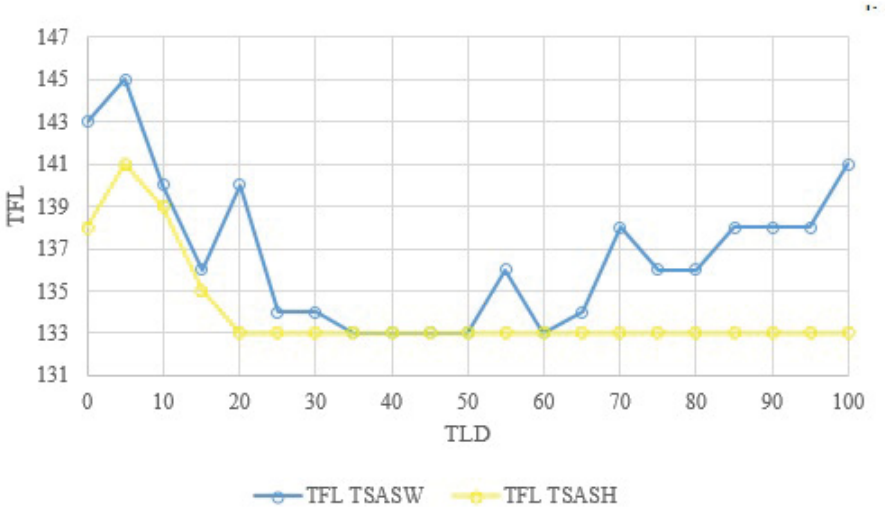

Figure 5.4 – Algorithm improvement over simulated TLD (first formulation of Austin 1996) with diversification.

The computation times required by our implementation of the TSA to solve Austin (1996) problem are presented in Table 5.17. Appendix B contains the detailed computation times of the three problems for each value of the tested tabu list duration.

Algoritm	*CPU time (sec.)*		
	Best	Average	Worst
TSASW	9.2899	9.6005	10.1160
TSASH	12.7261	13.2722	14.1694

Table 5.17 – Computation times used by the TSA procedures for the first formulation of Austin (1996) DSM

Finally, the most important results obtained in this section can be summarized in the last two columns of the Table 5.19.

Problem	**Size**	**Total Feedback Length**						
		Original Solution	Todd's Solution	Lancaster and Cheng's Solution	Quian's Solution	Optimal Solution	**TSASW Solution**	**TSASH Solution**
Kusiak (1991)	12	7	6	6	6	6	**6**	**6**
Steward (1981)	20	93	24	24	24	24	**24**	**24**
Austin (1996)	51	320	158	157	146	unknown	**133**	**133**

Table 5.19 – TFL computed using different methods and DSMs

It is interesting to observe the performance of the **TSASW** and **TSASH** using the second formulation of Austin (1996) DSM, with an initial TFL of 317. The parameter values are the same as those used in the former Austins (1996) DSM provided by Qian and Lancaster. The improvement graph for the **TSASW** and **TSASH** is found in Figure 5.5. The improvement curve shows that both **TSASW** and **TSASH** find a best solution sequence with a TFL of 137. The TFL of the second formulation is greater than the TFL of the first formulation; this can be explained by the fact that the second formulation has a greater initial TFL.

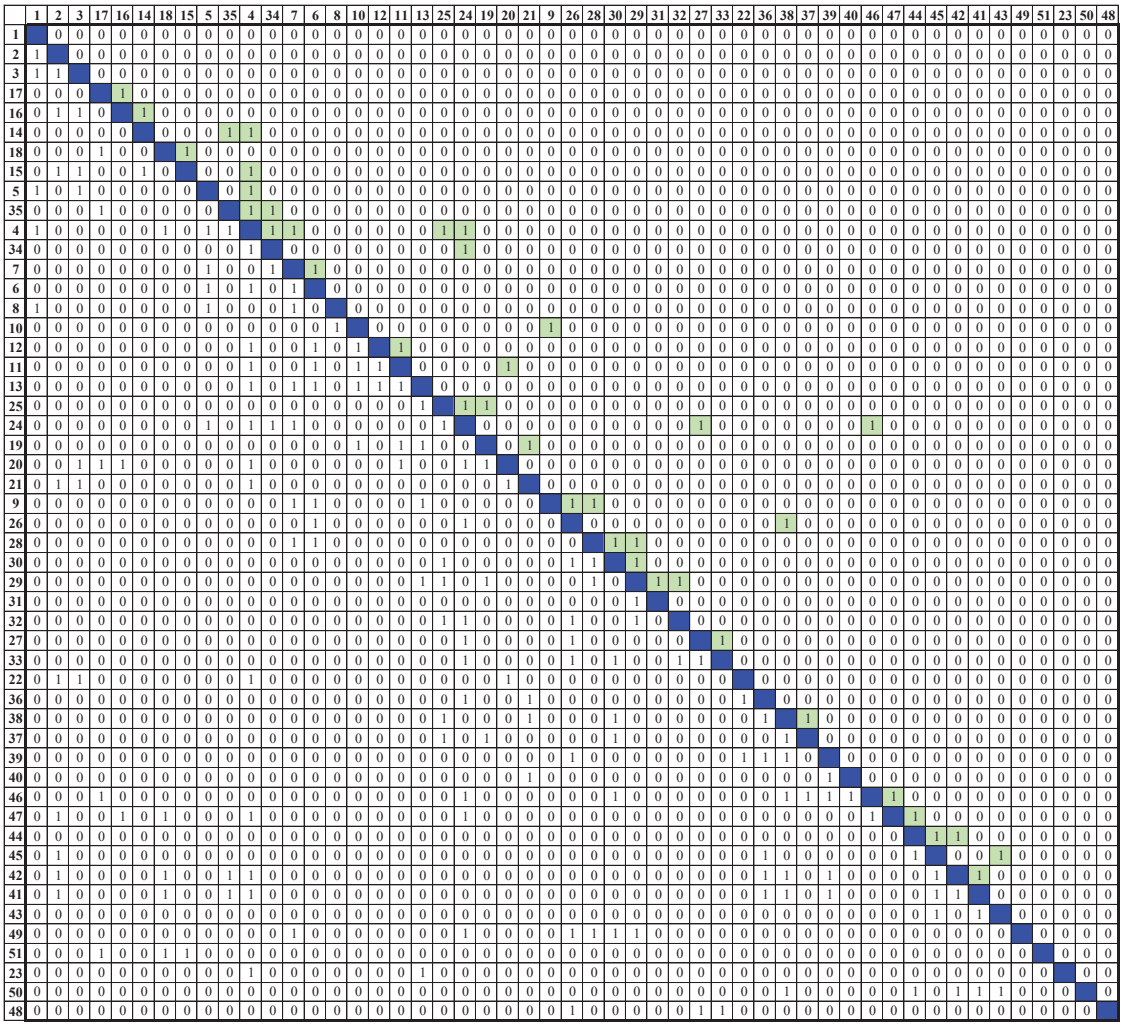

Table 5.18 – Reordered DSM of the first formulation of Austin (1996) DSM (TLF = 133)

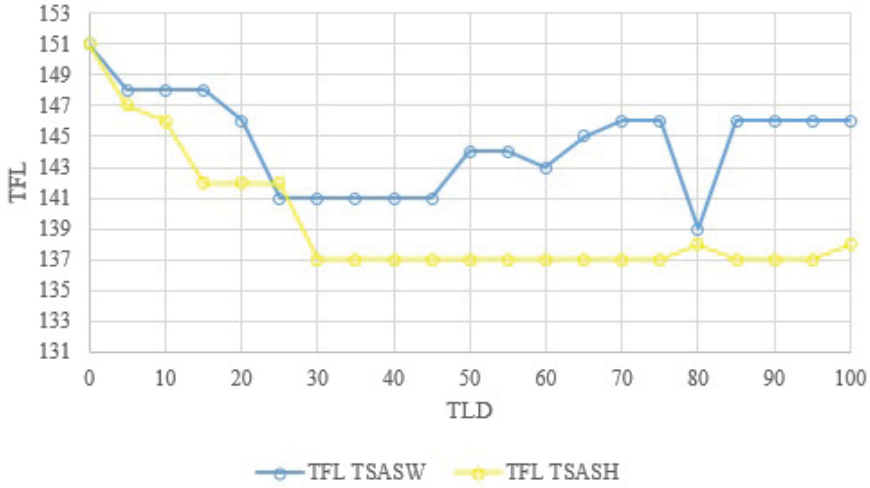

Figure 5.5 – Algorithm improvement over simulated TLD (second formulation of Austin 1996).

In addition, tests were also performed using a DSM matrix of 48 interrelated activities found in Qian and Lin [27]. In total, 70 activities are required to develop a balancing machine and 22 of the 70 activities were found to be independent, with a technique presented by Kusiak and Wang [21]. Since these activities can be performed simultaneously, they are not included in the DSM. Qian and Lin were not able to provide a numerical DSM because they could not estimate the degree of activity dependencies, due to a lack of historical data. Their model has two types of dependencies: "a soft dependence of activity *A* on activity *B* means that activity *A* depends on information input from activity *B*, but is allowed to precede activity *B*" and "a hard dependence of activity *A* on *B* means that activity *B* must precede activity *A*" [27].

Qian and Lin provide an initial matrix (see Figure 5.6) where the soft dependency is denoted by 1, and the hard dependency marked with "H" take a large value. This induces an initial solution having a total feedback length (TFL) equal to 358 ensuring that the hard dependency of activity *A* on *B* forces *B* to precede *A*. Qian and Lin indicate that as

long as "H" takes a value larger than 358, then the hard dependency is satisfied in the initial solution. Using their procedure, they obtain a solution having a value of 111.

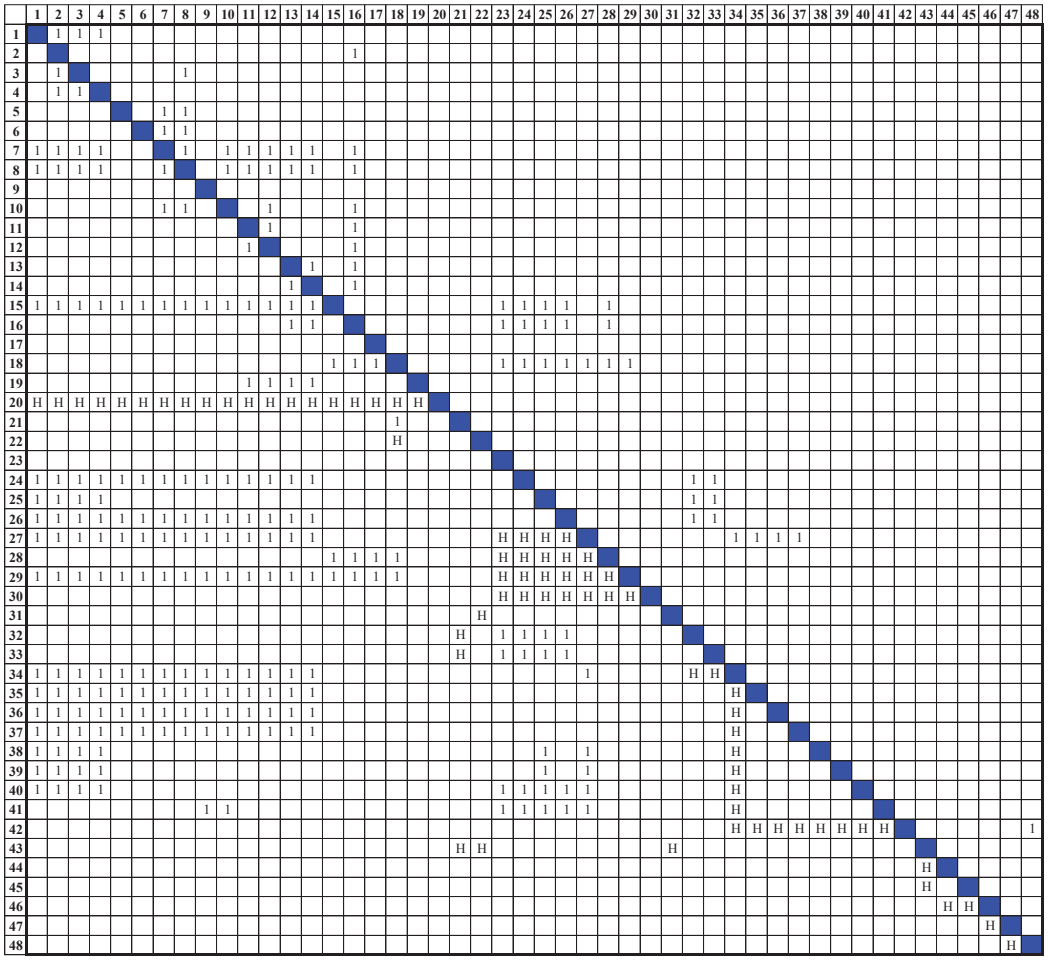

Table 5.20 – Unordered DSM of Qian (TFL = 358)

When we applied our heuristics to this DSM we obtained orderings with a TFL equal to 106 using the **TSASW** and equal to 107 using the **TSASH**. The stopping criterion was set to 1000 iterations according to Table 5.7 and the TLD was varied from 0 to 100, incremented by a factor of 5. The TSA improvement is presented in Figure 5.6 and the optimized DSM matrix and the solution sequence found by the TSA is in Table 5.21.

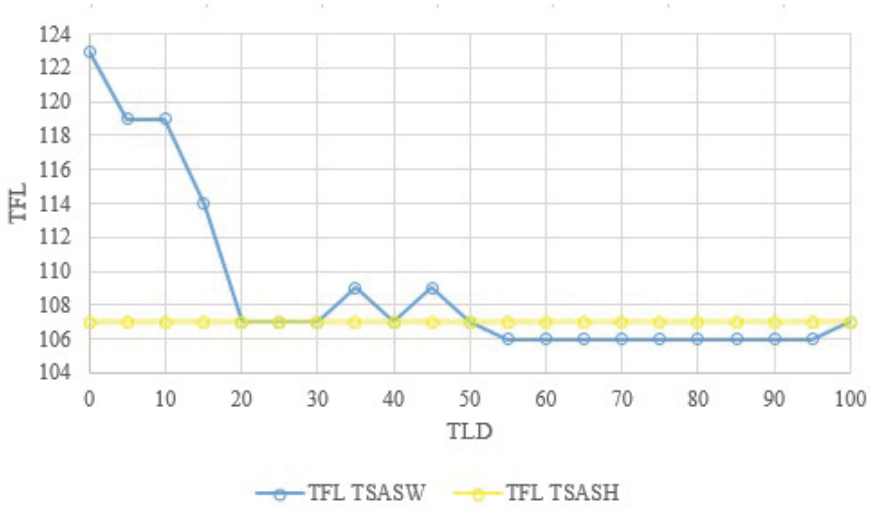

Figure 5.6 – Algorithm improvement over simulated TLD (Qian).

Both variants of our heuristic yield a TFL smaller than the one of 111 found by Qian and Lin using their heuristic.

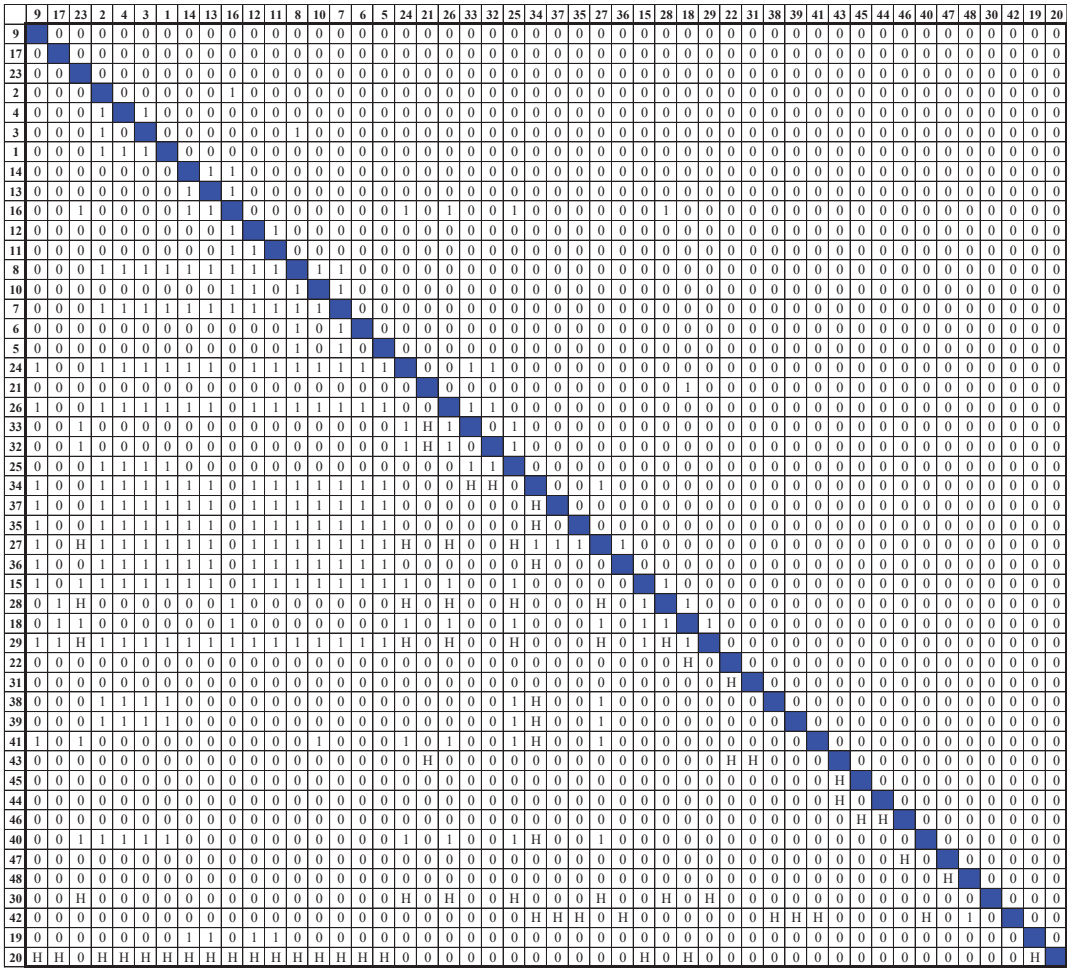

Table 5.21 – Reordered DSM of Qian (TFL = 106)

CHAPTER 6

CONCLUSION

In this paper, we study the feedback length minimization problem (FLMP) associated with a process design structure matrix (DSM), which is a $\mathscr{NP} - hard$ non-linear combinatorial problem that allows modeling complex interrelated activity scheduling problems. The objective function of the FLMP indicates a measure of the activity sequence disorder based on the information dependencies among activities that are interrelated. The overall TFL of the process activity sequence is proportional to the entropy of the activities.

We reduced Qian and Lin's formulation [27] of the total feedback minimization problem to a more simple and equivalent formulation. A new solution method based on the simple formulation was developed, that uses the tabu search to yield good solutions to the FLMP using two different neighborhoods : one based on a "swap" of two activities in the sequence and the other based on a "shift" of an activity to a new position in the sequence. The parameters of these two variants of the tabu search algorithm are tuned using problems of different sizes and densities. Afterwards, the tabu search algorithm is benchmarked with other methods found in the literature. Our method finds the optimal solution of the Kusiak (1991) and Steward (1981) problems. For the Austin (1996) problem, it outperforms the other methods found in the literature, yielding the best known solution for this problem. The TSA is lastly tested using a DSM that involves soft and hard dependencies of activities and again, our results outperform the results of the literature.

Two other types of DSM are less frequently used in product development : product architecture where the DSM elements are the components of the product and the interactions are the dependencies between the components and organization architecture where the DSM elements represent the people (or teams) and the interactions are the communications between them [7]. Nowadays, multi-domain DSMs are increasingly used to capture different aspects of project design, for example the people involved in the execution of the activities, the product to be developed, and the processes required to complete the project. Yassine [35] recently constructed a multi-domain objective function that minimizes the sum of the people, product and process single-domain costs. The proposed tabu search approach can be used to optimize the single-domain process cost of this objective function. By reducing the total feedback length of the single-domain process DSM, the tabu search algorithm can substantially decrease the total number of feedback marks located above the diagonal of the DSM and therefore can decrease the product domain cost.

The proposed tabu search algorithm provides a new strong base for the optimization of single-domain process DSMs which can also benefit to multi-domain DSM optimization. An interesting future research will be to use a hybrid approach, that combines the tabu search method with a genetic algorithm. The genetic algorithm can use adaptive parameters to store the best solution sequences found by the tabu search. The algorithm can apply crossovers by taking multiple best solution sequences found by the tabu search and recombining them to find better solutions. We have seen that each heuristic leads to a good solution by itself. To get a more robust and well-performing solution approach, we could also combine both heuristics presented in this paper.

BIBLIOGRAPHY

[1] H. M. Abdelsalam and H. P. Bao. A simulation-based optimization framework for product development cycle time reduction. *IEEE Transactions on Engineering Management*, 53(1):69–85, 2006.

[2] R. Ahmadi, T. A. Roemer, and R. H. Wangi. Structuring product development processes. *European Journal of Operational Research*, 130(3):539–558, 2001.

[3] S.A Austin, A.N. Baldwin, T. Hassan, and A.J. Newton. Techniques for the management of information flow in building design. In *Information Processing in Civil and Structural Engineering Design*, pages 119–123, Edinburgh, Scotland, 1996.

[4] T. .A. Black, C.H. Fine, and E.M. Sachs. *A method for systems design using precedence relationships: An application to automotive brake systems*. Sloan School of Management, Massachusetts Institude of Technology, 1990.

[5] K. B. Clark and T. Fujimoto. *Product development performance: strategy, organization, and management in the wold auto industry*. Harvard Business School Press, 1991.

[6] R. Durstenfeld. Algorithm 235 : random permutatin. *Communications of the ACM*, 7(7):420, 1964.

[7] S. D. Eppinger and T. R. Browning. *Design structure matrix methods and applications*. MIT Press, 2012.

[8] S.D Eppinger, D.E. Whitney, R.P. Smith, and D.A. Gebala. A model-based method for organizing tasks in product development. In *ASME conference on design theory and methodology*, pages 39–36, Chicaho, Illinois, September 1990.

[9] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. Gebala. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1):1–13, 1994.

[10] J. A. Ferland and D. Costa. Heuristic search methods for combinatorial programming problems. *Publication DIRO-1193, Department of Computer Science and Operations Research*, 2001.

[11] D. A. Gebala and S. D. Eppinger. *Methods for analyzing desingn procedures*. Sloan School of Management, Massachusetts Institute of Technology, 1991.

[12] M. Gendreau and J. -Y. Potvin. Tabu search. In *Handbok of Metaheuristics*, pages 41–59. Springer US, 2010.

[13] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and operations research*, 13(5):533–549, 1986.

[14] F. Glover and M. Laguna. Tabu search. In *Handbok of Combinatorial Optimization*, pages 3261–3362. Springer, 2013.

[15] D. L. Grose. Reengineering the aircraft design process. In *Proceedings of the 5th AIAA/USA/NASA/ISSMO symposium on multidisciplinay analysis and optimization*, pages 7–9, Panama City Beach, Florida, September 1994.

[16] J. Lin, Y. Qian, and W. Cui. Managing the concurrent execution of dependent product development stages. *IEEE Transactions on Engineering Management*, 59(1): 104–114, 2012.

[17] N. R. Joglekar, A .A.Yassine, S. D. Eppinger, and D .E. Whitney. Performance of coupled product development activities with a deadline. *Management Science*, 47(12):1605–1620, 2001.

[18] A. Karniel and Y. Reich. From DSM-based planning to design process simulation : a review of process scheme logic verification issues. *IEEE Transactions on Engineering Management*, 56(4):636–648, 2009.

[19] J. Kelley and M. Walker. Critical-path planning and scheduling : Mathematical basis. *Operations research*, 9(3):296–320, 1961.

[20] V. Krishnan and K. T. Ulrich. Product development decisions: A review of the literature. *Management Science*, 47(1):1–21, 2001.

[21] A. Kusiak and J. Wang. Efficient organizing of design activities. *International Journal of Production Research*, 31(4):753–769, 1993.

[22] J. Lancaster and K. Cheng. A fitness differential adaptive parameter controlled evolutionary algorithm with application to the design structure matrix. *International Journal of Production Research*, 46(18):5043–5057, 2008.

[23] J. Lin, Y. Qian, A. A. Yassine, and W. T. Cui. A fuzzy approach for sequencing interrelated activities in a DSM. *Int. J. Prod. Res*, 50:7012–7025, 2012.

[24] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operation Research*, 7 (5):646–669, 1959.

[25] C. Meier, A. A. Yassine, and T. R. Browning. Design process sequencing with competent genetic algorithms. *Journal of Mechanical Design*, 129(6):566–585, 2007.

[26] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[27] Y. Qian and J. Lin. Organizing interrelated activities in complex product development. *IEEE Transactions on Engineering Management*, 61(2):298–309, 2014.

[28] Y. Qian, J. Lin, T. N. Goh, and M. Xie. A novel approach to DSM-based activity sequencing problem. *IEEE Transactions on Engineering Management*, 58(4):668–705, 2011.

[29] J. L. Rogers, C. M. McCulley, and C. L. Bloebaum. *Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes*. Springer, 1996.

[30] R. P. Smith and S. D. Eppinger. A predictive model of sequential iteration in engineering design. *Management Science*, 43(8):1104–1120, 1997.

[31] R. P. Smith and S. D. Eppinger. Product development decisions: A review of the literature. *Management Science*, 43(1):1104–1120, 1997.

[32] D. V. Steward. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28(3): 428–442, 1981.

[33] D. Todd. *Multiple criteria genetic algorithms in engineering design and operation*. PhD thesis, Engineering Design Centre, University of Newcastle upon Tyne, UK, 1997.

[34] J. N. Warfield. Binary matrices in system modeling. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(5):441–449, 1973.

[35] A. A. Yassine, R. H. Chidiac, and I. H. Osman. Simultaneous optimisation of products, processes, and people in development projects. *Journal of Engineering Design*, 24(4):272–292, 2013.

Appendices

APPENDIX A

AVERAGE TOTAL FEEDBACK LENGTH

	Total Feedback Length (N=8)											
	Density											
	0.1		0.2		0.3		0.4		0.5		0.6	
TLD	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW
0	0.89	1.13	3.87	3.83	11.68	11.74	19.05	19.03	28.51	28.61	38.47	38.42
5	0.86	0.89	3.64	3.64	11.37	11.35	18.7	18.7	28.2	28.2	38.27	38.28
10	0.86	0.86	3.64	3.64	11.34	11.34	18.7	18.7	28.2	28.2	38.26	38.26
15	0.86	0.86	3.64	3.64	11.37	11.34	18.7	18.7	28.21	28.2	38.26	38.26
20	0.86	0.87	3.69	3.64	11.46	11.34	18.7	18.7	28.25	28.2	38.29	38.26
25	0.86	0.88	3.69	3.64	11.47	11.35	18.7	18.7	28.27	28.2	38.29	38.26
30	0.86	0.91	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
35	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
40	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
45	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
50	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
55	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
60	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
65	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
70	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
75	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
80	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
85	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
90	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
95	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26
100	0.86	0.94	3.69	3.65	11.47	11.39	18.7	18.7	28.27	28.2	38.29	38.26

	Total Feedback Length (N=16)											
	Density											
	0.1		0.2		0.3		0.4		0.5		0.6	
TLD	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW
0	9.11	8.03	51.67	51.45	105.66	105.02	167.42	167.52	236.13	236.42	316.87	316.93
5	7.96	7.69	50.2	49.59	104.4	104.1	166.39	166.03	235.53	235.55	315.6	315.98
10	7.51	7.51	48.95	49.41	103.57	103.76	165.42	165.64	234.93	235.17	315.16	315.17
15	7.4	7.47	48.63	48.84	103.22	103.65	165.14	165.53	234.61	234.83	314.95	315.09
20	7.34	7.4	48.74	48.68	103.22	103.35	165.18	165.24	234.64	234.65	314.8	314.81
25	7.38	7.39	48.77	48.66	103.24	103.22	165.3	165.1	234.77	234.59	314.87	314.78
30	7.42	7.36	48.85	48.63	103.43	103.2	165.42	165.1	234.74	234.64	314.88	314.76
35	7.44	7.35	48.92	48.61	103.45	103.18	165.4	165.04	234.83	234.54	314.99	314.75
40	7.49	7.31	49.09	48.68	103.58	103.01	165.41	165.05	234.9	234.55	315.03	314.76
45	7.5	7.31	49.14	48.55	103.67	103.03	165.51	165.05	234.97	234.57	315.1	314.75
50	7.58	7.31	49.28	48.46	103.75	103.02	165.58	165.06	235.07	234.54	315.16	314.72
55	7.61	7.31	49.24	48.46	103.92	103.02	165.53	165.08	235.03	234.56	315.15	314.73
60	7.67	7.31	49.33	48.51	103.96	103.02	165.68	165.1	235.08	234.59	315.24	314.8
65	7.66	7.30	49.25	48.54	103.84	103.05	165.62	165.09	235.14	234.59	315.24	314.81
70	7.68	7.31	49.46	48.53	103.92	103.06	165.7	165.13	235.08	234.62	315.28	314.73
75	7.7	7.31	49.46	48.53	103.89	103.07	165.62	165.12	235.23	234.6	315.31	314.76
80	7.77	7.31	49.44	48.53	103.92	103.06	165.7	165.13	235.21	234.63	315.32	314.83
85	7.73	7.31	49.44	48.51	104.02	103.09	165.78	165.1	235.25	234.61	315.33	314.78
90	7.74	7.32	49.5	48.56	104.03	103.1	165.78	165.14	235.23	234.65	315.34	314.81
95	7.75	7.32	49.54	48.5	104.02	103.09	165.78	165.14	235.25	234.61	315.34	314.78
100	7.77	7.31	49.54	48.53	104.03	103.11	165.78	165.14	235.23	234.65	315.35	314.77

	Total Feedback Length (N=30)											
	Density											
	0.1		0.2		0.3		0.4		0.5		0.6	
TLD	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW
0	118.73	116.78	449.5	446.33	823.4	823.56	1248.08	1249.18	1696.41	1694.34	2164.56	2164.76
5	117.13	115.11	447.21	444.86	821.72	820.24	1245.26	1245.46	1693.93	1691.54	2162.45	2162.99
10	115.25	113.11	445.89	442.59	820.03	818.42	1243.61	1244.17	1691.41	1690.3	2160.05	2160.36
15	113.71	112.44	444	442.06	819.25	817.54	1242.01	1242.31	1690.65	1689.71	2159.19	2158.83
20	112.64	111.55	441.73	441.39	817.54	816.9	1240.52	1241.54	1688.45	1688.63	2158.36	2158.03
25	111.89	111	440.74	441.18	816.69	816.52	1239.21	1241.19	1687.42	1688.3	2156.46	2157.43
30	110.65	110.75	440.33	440.38	815.93	816.47	1239.1	1240.3	1686.82	1687.55	2155.19	2156.41
35	109.78	110.46	439.89	440.05	816.19	816.24	1238.52	1239.76	1687	1686.8	2154.44	2155.08
40	110.16	110.1	440.44	439.35	816.28	816.13	1238.6	1239.24	1687.04	1686.71	2154.86	2154.97
45	110.41	109.34	439.82	439.23	816.44	815.96	1239.04	1238.58	1686.8	1686.21	2154.46	2154.64
50	109.81	109.51	440.07	438.51	816.87	815.83	1238.9	1237.93	1686.96	1686.09	2154.79	2154.85
55	109.95	109.35	439.85	438.24	816.92	815.65	1238.94	1237.82	1687.19	1685.61	2154.62	2154.57
60	109.87	109	439.94	438.38	817.02	815.8	1239.48	1237.39	1687.52	1685.52	2154.9	2154.27
65	110.44	109.07	440.21	437.83	817.1	815.45	1239.28	1237.58	1687.4	1685.28	2155.18	2154.21
70	110.43	108.96	440.35	437.78	817.42	815.32	1239.73	1237.31	1687.55	1684.97	2155.22	2154
75	110.63	108.84	440.68	437.74	817.61	815.49	1239.67	1237.08	1687.56	1685.2	2155.46	2154.07
80	110.73	108.74	441	437.51	817.45	815.5	1240.02	1237.12	1687.68	1684.93	2155.69	2153.78
85	111.1	108.75	441.23	438	817.75	815.34	1239.96	1237.04	1687.94	1685.07	2155.6	2153.81
90	111.26	108.57	441.12	437.54	817.91	815.17	1240.04	1236.84	1688.21	1685.44	2155.73	2153.77
95	111.16	108.78	441.17	437.44	817.93	815.11	1240.37	1237.33	1688.24	1685.12	2156.08	2153.45
100	111.53	108.7	441.57	437.58	817.82	815.07	1240.25	1237.21	1688.33	1684.93	2155.78	2153.46

	Total Feedback Length (N=60)											
	Density											
	0.1		0.2		0.3		0.4		0.5		0.6	
TLD	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW	TSSH	TSSW
0	1606.1	1599.2	4521.7	4519	7705.4	7695.2	11008.4	10999.4	14510	14524.2	18413.5	18410.6
10	1603.1	1597.6	4513.7	4516.3	7702.8	7678.7	10999.3	10995.8	14504	14514.6	18412.2	18397
20	1596.4	1592.5	4509.1	4508.4	7695.5	7671.8	10985.5	10987.7	14501.9	14510	18411	18392
30	1593.1	1591.5	4501.4	4506.1	7687	7669.7	10976.7	10987	14493.9	14508.2	18407.4	18388.1
40	1593.2	1588.7	4500.8	4494.2	7680.7	7665.7	10965.2	10985.4	14494.1	14509.2	18404.2	18388
50	1576.4	1588.3	4497.7	4493.2	7673.7	7666.8	10956.8	10979.9	14493	14503	18387.9	18387.4
60	1575.8	1587.9	4493.3	4490.1	7676	7665.9	10952.9	10978.9	14494.2	14485.1	18376.7	18387.3
70	1576	1586.6	4484.6	4485.5	7672.8	7664.5	10953	10953.6	14494.3	14484.5	18371.3	18384.4
80	1572.8	1586.4	4486.3	4482.5	7671.4	7663.7	10952.4	10976.9	14493.4	14483.1	18367.5	18384.6
90	1573.9	1586.4	4484.9	4483.6	7673.4	7662.9	10954.7	10950.2	14494.2	14477.8	18369.1	18384.6
100	1571.7	1582.8	4482.9	4478.1	7670.2	7662.4	10942.3	10946.4	14489.6	14477.7	18369.4	18382.6
110	1570.7	1586.2	4486	4484.6	7673.8	7653.4	10939.7	10947.1	14494.5	14477.5	18361	18377.2
120	1572.7	1579.6	4482	4479.7	7678.9	7653	10941.9	10938.6	14484.7	14474.5	18363	18374.4
130	1571.8	1574.5	4480.1	4477.3	7659.7	7653.5	10943.1	10945.1	14492.1	14472.6	18361.2	18380.6
140	1574.2	1569.5	4481.6	4477.6	7663.7	7656.7	10945.2	10934.4	14493.6	14475.1	18360.6	18377.3
150	1576.5	1575.3	4482.9	4480.9	7676.8	7653	10942.9	10933.8	14483.5	14473.1	18357.9	18373.7
160	1571.8	1572.5	4482.7	4474.1	7661.1	7652.6	10944.5	10933.9	14492.7	14474.9	18360.3	18373.9
170	1572.9	1573	4483.9	4474.1	7673.6	7651.8	10947.7	10934.7	14481.2	14474.8	18359.8	18374.8
180	1576.8	1567.9	4481.8	4475.2	7665.5	7653.4	10946.2	10933.8	14482.4	14475.3	18364	18376.9
190	1574.4	1569.9	4488.3	4475.6	7667.2	7654.8	10950.6	10933.6	14486.7	14474.7	18362.9	18373.5
200	1573.6	1569.3	4486.3	4474.5	7667	7653.2	10950.1	10933.7	14481.6	14473.4	18359.1	18374.3

	Total Feedback Length (N=120)											
	Density											
	0.1		0.2		0.3		0.4		0.5		0.6	
TLD	**TSSH**	**TSSW**	**TSSH**	**TSSW**	**TSSH**	**TSSW**	**TSSH**	**TSSW**	**TSSH**	**TSSW**	**TSSH**	**TSSW**
0	16775.6	16745.4	41438.0	41308.2	68103.2	67872.2	95716.2	95661.6	124272.2	124197.4	153816.2	153725.4
20	16745.6	16698.2	41376.2	41299.0	68076.2	67787.6	95680.4	95658.0	124265.0	124191.8	153808.6	153708.6
40	16718.0	16634.0	41417.8	41284.0	68053.0	67757.4	95665.4	95596.4	124208.8	124170.8	153716.2	153618.2
60	16701.2	16625.6	41361.4	41280.4	68032.4	67780.8	95664.6	95595.0	124206.4	124169.0	153687.8	153611.6
80	16698.8	16600.0	41334.6	41280.0	68019.0	67763.4	95626.4	95590.8	124161.8	124162.4	153687.8	153617.8
100	16703.2	16602.4	41323.0	41280.0	67977.0	67755.8	95625.2	95587.2	124131.4	124160.4	153710.2	153617.6
120	16676.6	16625.2	41330.8	41280.0	67927.8	67754.4	95629.6	95585.8	124119.8	124168.8	153645.0	153606.8
140	16662.8	16620.0	41296.0	41280.0	67883.2	67748.0	95625.2	95585.6	124063.6	124158.6	153646.2	153611.6
160	16652.2	16620.0	41302.0	41279.6	67923.4	67719.4	95625.6	95571.2	124053.2	124155.2	153597.4	153611.8
180	16640.4	16613.2	41289.8	41272.6	67889.0	67735.4	95607.6	95559.6	124047.2	124138.8	153558.8	153611.6
200	16589.2	16586.0	41264.8	41271.8	67938.8	67738.6	95624.0	95558.4	124080.0	124117.2	153559.8	153601.2

APPENDIX B

COMPUTATION TIMES OF THE KUSIAK (1991), STEWARD (1981) AND AUSTIN (1996) PROBLEMS

TLD	*CPU time (sec.)*					
	Kusiak (1991)		**Steward (1981)**		**Austin (1996)**	
	TSASW	**TSASH**	**TSASW**	**TSASH**	**TSASW**	**TSASH**
0	0.0061	0.0067	0.0888	0.1211	9.6840	13.2239
5	0.0057	0.0067	0.0861	0.1181	9.6287	13.9749
10	0.0055	0.0067	0.0820	0.1155	10.0241	14.1694
15	0.0060	0.0064	0.0824	0.1178	9.8649	13.2238
20	0.0058	0.0063	0.0823	0.1187	9.8686	13.6943
25	0.0058	0.0063	0.0987	0.1354	9.7281	13.6483
30	0.0058	0.0061	0.0939	0.1176	9.5208	13.8762
35	0.0055	0.0063	0.0819	0.1123	9.9308	13.5098
40	0.0055	0.0061	0.0838	0.1118	9.5247	13.2584
45	0.0055	0.0061	0.0833	0.1193	9.9600	13.9224
50	0.0012	0.0055	0.0955	0.1343	10.0886	13.9630
55	0.0012	0.0059	0.0889	0.1076	10.1160	13.1300
60	0.0011	0.0054	0.0825	0.1083	9.2923	12.8526
65	0.0012	0.0060	0.0848	0.1074	9.2979	12.8192
70	0.0012	0.0052	0.0820	0.1118	9.2899	12.8168
75	0.0011	0.0055	0.0983	0.1291	9.2944	12.7967
80	0.0011	0.0028	0.0917	0.1049	9.2988	12.7675
85	0.0011	0.0022	0.0832	0.1044	9.2922	12.7880
90	0.0011	0.0024	0.0834	0.1061	9.3000	12.7536
95	0.0011	0.0021	0.0825	0.1075	9.2956	12.7261
100	0.0011	0.0022	0.0980	0.1281	9.3107	12.8015