

# Manufacturing Production Line Modelling and Classification of Associated NP-Hard Problems

Joseph Bonello\*, John Abela and Ernest Cachia

University of Malta, Faculty of ICT, Msida, Malta

## Abstract

Optimisation of production lines is known to be NP-Hard in the general case so many near-optimal approximation algorithms have been researched to overcome the challenge [1]. In this paper we describe an approach to modelling production lines using a graph theoretic model. In particular, we focus on single machine and job shop problems. We show that the model can be extended to open shop problems. We also discuss how the model can be used to classify scheduling problems from the generated models.

## Publication History:

Received: November 27, 2017

Accepted: February 17, 2018

Published: February 19, 2018

## Keywords:

Scheduling, Graph-theoretic modelling, Production line modelling, Classification of computational complexity of scheduling problems

## Introduction

Scheduling in manufacturing is an area that has interested researchers for at least the past 60 years [2]. It is the process of generating a plan (a schedule) that shows the order in which tasks are to be executed in order to achieve a purpose [3]. Together with planning, it lies at the heart of important decision processes in manufacturing companies.

These processes affect the central operations of the company, including procurement, production, distribution, transportation, information processing and communication [2]. Maximising productive manufacturing time is essential for the manufacturing industry to maintain profitability [4]. This requires efficient utilisation of resources as outlined by Edgett in his article for Manufacturing.net, a leading web site with up-to-date information about manufacturing [5]. And better scheduling ensures the best employment of available resources.

There is an outstanding amount of research related to scheduling problems. Blazewicz et al. define scheduling problems as the problem of allocating resources over time to perform a set of tasks that make up a larger process [6]. Individual tasks compete for resources and tasks can have relations between them affecting the way in which they can be processed.

Today, the manufacturing industry is facing very difficult market conditions that force it to strive for efficiency besides guaranteeing quality. This has given rise to the term Lean Manufacturing. This is a production control technique that aims at eradicating inefficiencies from the manufacturing process. There are a number of techniques that build on this philosophy, namely the Toyota Production System, Just-In-Time (JIT) Manufacturing, Kanban and 5S [7].

Scheduling theory deals with different problem types, and a large volume of literature, models and approaches that describe and attempt to solve different scheduling problems. Scheduling theory uses information already at hand at the manufacturing plant to find a feasible schedule for the manufacturing operations to be executed on/by the required resources. This information includes, but it is not limited to, the type and amount of each resource used in the manufacturing process, the sequence of the tasks necessary to manufacture an item and the time required for each task [3].

## The purpose and scope of this study

This study was part of a two-year research project financed by the European Regional Development Fund (ERDF) called Research Services in Manufacturing: ICT in Manufacturing. The objective of this process was to create a software solution that allows operators to model production lines and determine and classify the scheduling problems represented by the model.

It has long been felt that not enough information is available regarding the scheduling problems affecting the manufacturing industry in Malta. This project was undertaken to gain additional information on the state of scheduling in the local manufacturing industry and to provide an extensible framework that can be used for current analysis and future research.

This study focused on providing a graphical means of representing scheduling problems. This representation describes the abstraction of the production line involved in manufacturing a product. The second aim of the study is to create a set of heuristics using the model to classify the scheduling problem according to its computational complexity, and when possible, pointing to the relevant literature used to derive the classification in order to help researchers understand the problem better.

Scheduling involves efficient utilisation of scarce resources. While this study focuses on scheduling in manufacturing industry, Leung draws a parallel to the problems encountered by computer scientists in the 1960s when computational resources (CPU, memory and I/O devices) were scarce [8]. This results of this study are of interest to computer scientists, operators in the manufacturing industry whose work deals with scheduling and to researchers in the field of optimisation.

**Corresponding Author:** Joseph Bonello, University of Malta, Faculty of ICT, Msida, Malta; E-mail: [joseph.bonello@um.edu.mt](mailto:joseph.bonello@um.edu.mt)

**Citation:** Bonello J, Abela J, Cachia E (2018) Manufacturing Production Line Modelling and Classification of Associated NP-Hard Problems. Int J Comput Softw Eng 3: 128. doi: <https://doi.org/10.15344/2456-4451/2018/128>

**Copyright:** © 2018 Bonello et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

## Production Line Graph Modelling

Graphs are a flexible mathematical structure that are used for representing a wide variety of problems. In particular, they are useful for representing relationships between abstract entities. Bondy and Murty, in fact, note that many real-world situations can be modelled using graphs [9].

Bondy and Murty define a graph  $G$  as an ordered pair  $(V(G), E(G))$  consisting of a set  $V(G)$  of vertices (or nodes) and a set  $E(G)$  disjoint from  $V(G)$ , of edges with an incidence function  $\phi_G$  that associates with each edge of  $G$  an unordered pair of vertices of  $G$  [9]. If  $e$  is an edge and  $u$  and  $v$  are vertices such that  $\phi_G(e) = \{u, v\}$  then  $e$  is said to join  $u$  and  $v$ .  $u$  and  $v$  are called the ends of  $e$ . The number of vertices in  $G$  is denoted by  $v(G)$  and the number of edges is denoted by  $e(G)$ .  $v(G)$  and  $e(G)$  are referred to as the order and the size of  $G$  respectively [9].

Graphs have been used in many of areas of computer science. For instance, Heller and Schneiderman describe how data structures can be modelled using graph theoretic principles [10]. Their paper describes how graph-based data structures can be used in databases for organising records supporting the search for data in a given field or a subset of fields. Mirza has used a graph theoretic modelling approach to link content in a recommender system [11]. The aim of the algorithm was to use graph theory to identify links between entities in a social network. Caramuta used a graph-theoretic approach to Decision Theory in order to explicitly represent memory when individuals make choices [12].

There are several examples of how a graph-theoretic model has been employed to represent aspects of scheduling problems. Sanmarti et al. use a graph representation called the S-Graph for specifying specific chemical processes in multipurpose batch plants [13,14]. Their approach demonstrates how graphs are well suited for modelling complex scenarios as in this case they model a flexible environment where the number of products is very large and where alternative paths may be used to manufacture the same product.

A second technique described by Friedler and Fan for modelling a manufacturing process is the P-Graph [15]. Also known as the Process Graph, this is a bipartite representation of the structure of a process system. Operating units in P-Graph are represented by horizontal bars while input and output materials are represented by solid circles. The direction of the edges of the P-graph indicate the flow of material in the network. Operating units cannot be connected, and therefore, there cannot be any edges between horizontal lines.

P-graphs were designed to overcome problems in graph-based approaches depending on directed graphs or signal flow graphs. In digraph modelling, operating units correspond to vertices while connections are represented by edges of the graph. In signal-flow graphs, vertices represent the materials of the process. While these models excel at representing and analysing the system, they fail to adequately describe process synthesis. The sleek form of the P-graph captures the syntactic and semantic contents of the process structure.

Two other graph representations that have been considered are the Resource-Task Network (RTN) and the State Task Network (STN). The first was developed by Pantelides and is a unified framework that is used in the description and solution of a variety of process scheduling problems [16]. TRTN is based on bipartite graphs, and is made of two nodes:

1. **Asks:** A task is an operation that transforms a certain set of resources into another set.
2. **Resources:** A resource includes all entities that are involved in the process such as materials, processing and storage equipment, utilities and setup conditions.

The State-Task Network (STN) is a scheduling model that is used in multi-product batch plants and that can be applied to batch processes that are specified by recipes. It is used in short-term scheduling where demands for products are specified by deadlines [17]. The capabilities of STN are:

1. The model does not require a fixed assignment of equipment to process tasks;
2. Batches are variable-sized, and can be mixed and split;
3. The model supports different intermediate storage and transfer policies as well as limitations of resources.

## The challenge of modelling production lines

The challenge of modelling production lines is two-fold:

1. Exhaustively describe the production process; the graph must represent all the essential elements of the production line unambiguously and the relationships between the different elements of the production line. Moreover, it should enable the viewer, or an algorithm, to infer information about the production line.
2. Comprehensive and easy-to-use by a non-technical operator; Often operators in scheduling are not conversant in Computer Science or Operations Research, hence it is necessary that the model produced is intuitive for non-specialised operators. A simple model describing the production line allows operators who are familiar with the production line to verify the models and ensure that it faithfully represents the different elements of the production line.

The advantage of a simple graph is useful not only for operators, but it allows the practitioner to quickly get an overview of the production line and formulate a pre-defined signature of the problem(s) present in the production line. Graphical representation using graphs allows inspections on the relationships represented by the graph.

## Defining Scheduling Problems

Formal models based on this information allow researchers to define and understand scheduling problems and then search for optimal solutions. These models are required to remove ambiguity, to arrive at an improved descriptive precision [18]. Such methods allow for consistency, analysis and validation, thus ensuring correctness. They can be used for working out the consequences of particular constraints, and also help to identify the effects of possible changes.

We refer to the notation by Graham et al. to describe a scheduling problem [19]. This notation is widely used in literature and should provide a coherent platform for discussing the composition of scheduling problems. In summary, this notation describes scheduling problems using three fields, namely  $\alpha$ ,  $\beta$  and  $\gamma$  described hereunder:

$\alpha$  refers to the machine environment and is made up of two fields,  $\alpha_1$  and  $\alpha_2$ .  $\alpha_1$  represents the machine environment and can be an element of  $\{\epsilon, P, Q, R, F, J, O\}$ . The following list describes the different values that can be used in each of the fields:

1.  $\alpha_1 =$  there is a single machine,
2.  $\alpha_1 = P$  the machine environment consists of identical machines,
3.  $\alpha_1 = Q$  the machine environment consists of uniform machines,
4.  $\alpha_1 = R$  the machine environment consists of unrelated machines
5.  $\alpha_1 = F$  a flow shop represents a problem in which jobs have one operation for each machine such that all machines are used in the same order;
6.  $\alpha_1 = J$  a job shop is a problem where each operation is ordered according to some manufacturing plan;
7.  $\alpha_1 = O$  a open shop is a problem where there are no constraints on operation processing.

$\alpha_2$  represents the number of machines (given as a positive integer). This field can store the values  $\{\epsilon; 1, 2, 3, 4, 5, m\}$ .  $\alpha_2$  can be 1 only if  $\alpha_1 = \epsilon$ . When  $\alpha_2 = \epsilon$ , there is a variable number of machines.

$\beta = \{\beta_1; \beta_2; \beta_3; \beta_4; \beta_5; \beta_6; \beta_7; \beta_8\}$  represent the job constraints of the scheduling problem.

1.  $\beta_1 \in \{pmtn; \epsilon\}$  - This field represents job-splitting or pre-emption;
2.  $\beta_2 \in \{res; res1; \epsilon\}$  - This field represents resource constraints that are required for the manufacturing of a job;
3.  $\beta_3 \in \{prec; tree; chains; \epsilon\}$  - Jobs may have precedence constraints defined on them, such that if a job depends on another it cannot start unless the job it depends on has been manufactured;
4.  $\beta_4 \in \{r; \epsilon\}$  - Jobs may have release dates, which are represented in these fields;
5.  $\beta_5 \in \{mj \leq \epsilon\}$  - Defines a ceiling on the performance of a machine's ability to manufacture the job;
6.  $\beta_6 \in \{p_i = 1; p_i = p; p_{ij} = 1; p_{ij} = p; \epsilon\}$  - Represent the processing times it takes to manufacture the job;
7.  $\beta_7 \in \{d; \epsilon\}$  - Jobs may have deadlines which are end dates beyond which the job cannot be manufactured;
8.  $\beta_8 \in \{no-wait; \epsilon\}$  - This field is present in Flow Shops, where no buffers are present and jobs are started in succession;
9.  $\beta_9 \in \{s-batch; p-batch; \epsilon\}$  - Represents a batching problem, a problem where jobs are manufactured jointly on one machine.

The optimality criteria for a schedule is given by  $\gamma \in \{f_{max}, \sum f_j\}$  and it represents the function to optimise. For a schedule  $s$ , we can compute for each job  $J_j$ :

1. the completion time or makespan  $C_j$ : the amount of time required to complete a pre-identified group of jobs;
2. the lateness  $L_j = C_j - d_j$ : the difference between a job's completion date and its due date;
3. the tardiness  $T_j = \max\{0; C_j - d_j\}$ : the amount of time it takes to complete a job once its due date has passed, otherwise 0;
4. the earliness  $E_j = \max\{0; d_j - C_j\}$ : the amount of time until a job's due date arrives once the job has been completed; or
5. the unit penalty  $U_j =$  if  $C_j \leq d_j$ , then 0 else 1: assign a penalty if the job was manufactured earlier than the due date.

Thus, the optimality criterion seeks to minimise  $f_{max} \in \{C_{max}, L_{max}\}$  where  $f_{max} = \max_j \{f_j(c_j)\}$  with  $f_j(C_j) = C_j, L_j$  or

$$\sum f_j \in \{\sum C_j, \sum T_j, \sum U_j, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$$

The optimal value of  $\gamma$  will be denoted by  $\gamma^*$  which is the value produced by an approximation algorithm  $A$  for  $\gamma(A)$ .

## Classification of Scheduling Problem

The classification of scheduling problems is done through a tool which classifies scheduling problems by matching them against a database of known results. The results are published and maintained by Knust and Brucker and include a set of digraphs that represent a reduction that are used to determine if one problem is related to another [20]. The principle of the classification engine is based on the model proposed by Lageweg et al [21], [22]. An online version of the tool was developed by Dürr [23].

Lageweg et al distinguish between "easy" and "hard" problem types by specifying that an "easy" problem is a problem for which there exists a polynomial time algorithm [21]. A "hard" problem is a problem which is known to be NP-Hard. MSPCLASS, the program they came up with, maintained a list of problems known as "easy" and "hard" problems. It then, systematically, employs a *partial ordering* to derive the status of the individual problem and to classify the problems as *open, easy or hard*. Moreover, it can also determine the following subclasses of problems, namely:

1. maximal easy problems;
2. minimal open problems;
3. maximal open problems;
4. maximal hard problems.

A particular problem may belong to different categories if its word list is a subset of the words of more than one category. The universe of a category is constructed using the union of the words in the components that are associated with the category [20]. A component is a directed acyclic graph whose nodes are word sets. The set of nodes of the DAG are the domain of the component. These components are the basic block of the reduction rules that the classification engine uses to deduce whether a problem is an instance of another. The union of the word sets of the components (which can belong to just one category) make up the universe of that category [20].

The graph of a component is connected and its transitive closure defines a partial order which specifies the rules to use to deduce the relationship between a particular problem and another problem in the database. The partial order is only valid within a certain context, which is the category. Figure 1 shows the reduction graph for  $\gamma$ , which is common to all categories. The comparison is only done when a problem falls within one of the components defined by the engine. Problems from different categories cannot be compared. If there is a category such that for every component in the category one problem  $\Pi_1$  has smaller or equal values than problem  $\Pi_2$  according to the partial order, then  $\Pi_1$  is a particular case of  $\Pi_2$ .

To determine the category of a problem, the classification engine finds the intersection of the category's universe with the word set of the problem. If the result is that the problem's word set is in the universe of the problem, and it has a value for every component associated with

the category, then the problem belongs to the category. This allows the classification engine to determine which problems can be matched to the problem being classified [21].

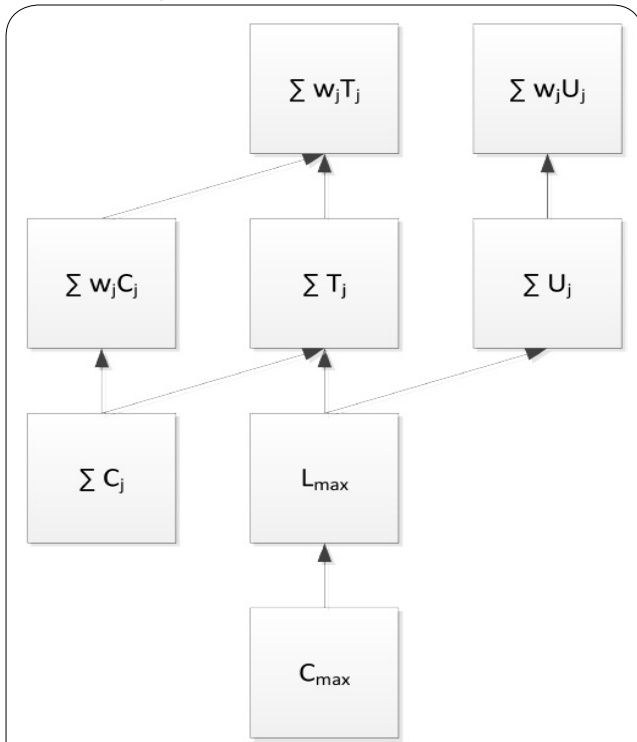


Figure 1: Reductions for  $\gamma$ , applying to all categories

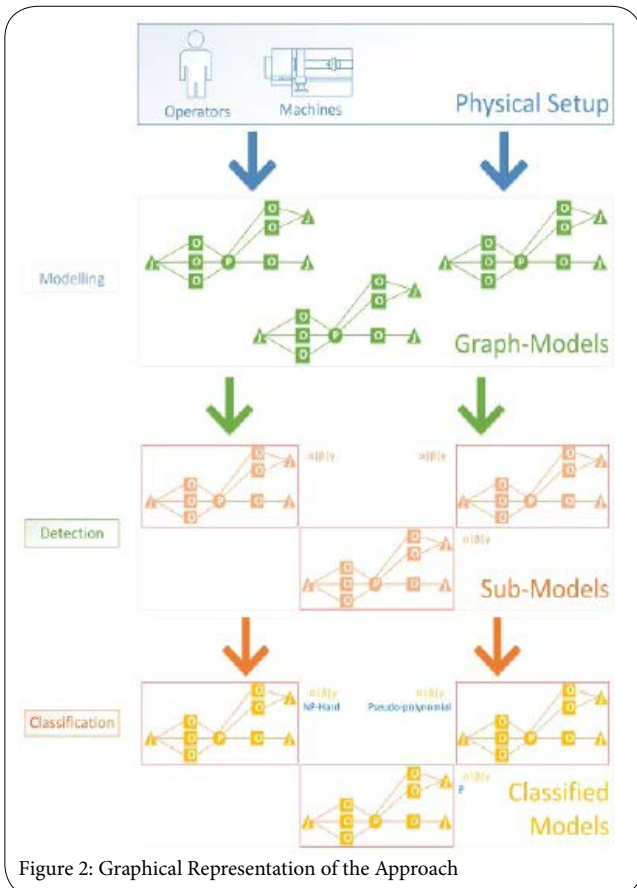


Figure 2: Graphical Representation of the Approach

The classification engine was developed in T-SQL on Microsoft SQL Server 2008 R2. The following are the rules that are used to determine whether one problem is a special case or a generalisation of another problem. The basic component of the classification engine is the *category*. A category represents a collection of problems, and it is defined as the collection of words that describe the collection of problems. Words are the elements that make up the signature of a problem. For example, considering  $1|pmtn; r_i| C_{max}$  as an example, the words for this signature would be 1,  $pmtn$ ,  $r_i$  and  $C_{max}$ . The union of the words in a category is called the *universe* of the category.

### Approach

The approach adopted in this study was to create a system whereby a user can “translate” a physical model into an appropriate abstraction that can be used by a computer to detect and classify scheduling problems. Figure 2 describes the process. The signatures were generated using the Graham et al notation are able to describe specific scheduling problems as well as classes of scheduling problems which are useful for classifying scheduling problems [19].

While the approach is good for a short technical description of a scheduling problem, it requires a good knowledge of the notation and know-how in analysing manufacturing shops and production lines. Our approach to model production lines builds upon the notation but uses a graph to describe the interactions between the different components. The nodes used are specified in table I.

Each graph node represents an entity of the  $\alpha|\beta|\gamma$  representation. An additional node represents information about the model. The model allows lines to be associated to different shops, as is the case in most manufacturing establishments. It is common terminology in manufacturing environments to refer to areas of a factory where manufacturing takes places as a shop floor.

A processor is a unit of manufacturing that is used to produce a product or a component. A processor is available to just one shop and may represent a single machine or a set of (parallel) machines. In accordance to the definitions proposed by [19], we distinguish between identical, uniform and unrelated parallel machines and further specify if the machines are multi-purpose machines.

Manufacturing jobs are not limited to be manufactured in a single shop. The operations (or tasks) of a job may be processed by machines in different shops. Hence a job represents the product that it related to and the operations represent the tasks that are required to produce the item. A job node gives information on the processing time of the job and gives details on whether the job has deadlines or release dates associated with it.

Job operations are associated primarily with the job to which they are related, and the machine(s) where they will be executed. Associated with the operation, there is also the individual processing time (which contributes to the total processing time of the job) and whether the job can be split after the operation has been completed. It is assumed that a job's operation is atomic, and that job splitting is permitted only after a whole operation has been concluded.

Figure 3 is an example of a simple production line used to test the modelling of a production line. Each node visually shows the information about the entity it represents. Internally, the node stores an entity object that contains a copy of the data that defines it. The model is persisted to the back-end database so that it can be reused. Table I provides a list of the information stored in each node type.



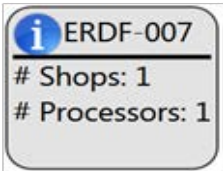
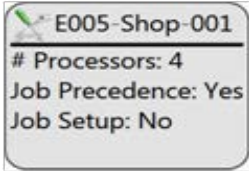
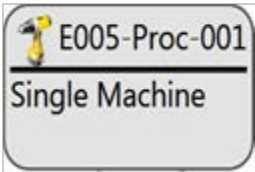
Node Type	Stored Information	Node Type	Stored Information
Example Node		Example Node	
Model	Model ID	Shop	Shop ID
	Model Name		Shop Name
	Model Description		Shop Description
	Model Data		Optimisation Function
			
Operation	Operation ID	Job	Job ID
	Job Name		Job Name
	Operation Order		Job Processing Time
	Processing Time		Related Product
	Related Subcomponent		Number of Tasks (Operations)
	Job may pause after operation (pre-emption)		Deadlines
	Machines where operation can be executed		Release Dates
Processor	Processor ID		
	Single Processor/Multiple Processor		
	Processor Grouping, Identical, Uniform or Unrelated machines specification		
	Maximum Production for Processor		
	Batching		
	Availability per day		
	Available days per week		
			

Table 1: Node Representation

## Evaluation

The evaluation was carried out in conjunction with local industry partners. A number of use cases were carried out showing how different production lines can be modelled and analysed using this tool.

### Use case 1

The partner in the first use case produces animal feeds for the local industry. The company has two production lines which work separately and the feed that is produced in one line cannot contaminate the other line. The use case focuses on the main production line, which is used most of the time. The company has 22 products that are produced on this line, and each product is represented by a single operation on the main line. An overview of the process is given in figure 4.

The lines were imported into the tool as shown in figure 5. The classification tool was run on the model and the classification shows that the process is a single machine problem with a known maximal polynomially solvable solution as shown in figure 6.

### Use case 2

The second example involves a partner whose product is manufactured using 4 machines in a direct line and 3 operators. The machines produce the individual components that are required for the product while the operators assemble the individual parts into the finished component. An overview of part of the process is given in figure 7.

In this scenario, since all the processing is done in sequence, the model is represented as a single machine in the model. This use case indicates that the capacity of the flowline is of 1,200 units per hour, thus indicating that each unit requires 3 seconds to manufacture.

An interesting feature of this use case is the presence of setup times between different jobs. Changing from one job to another requires changing large moulds that are used in the injection moulding process. To replace the mould, calibrate and test the machines, the process takes around 48 hours.

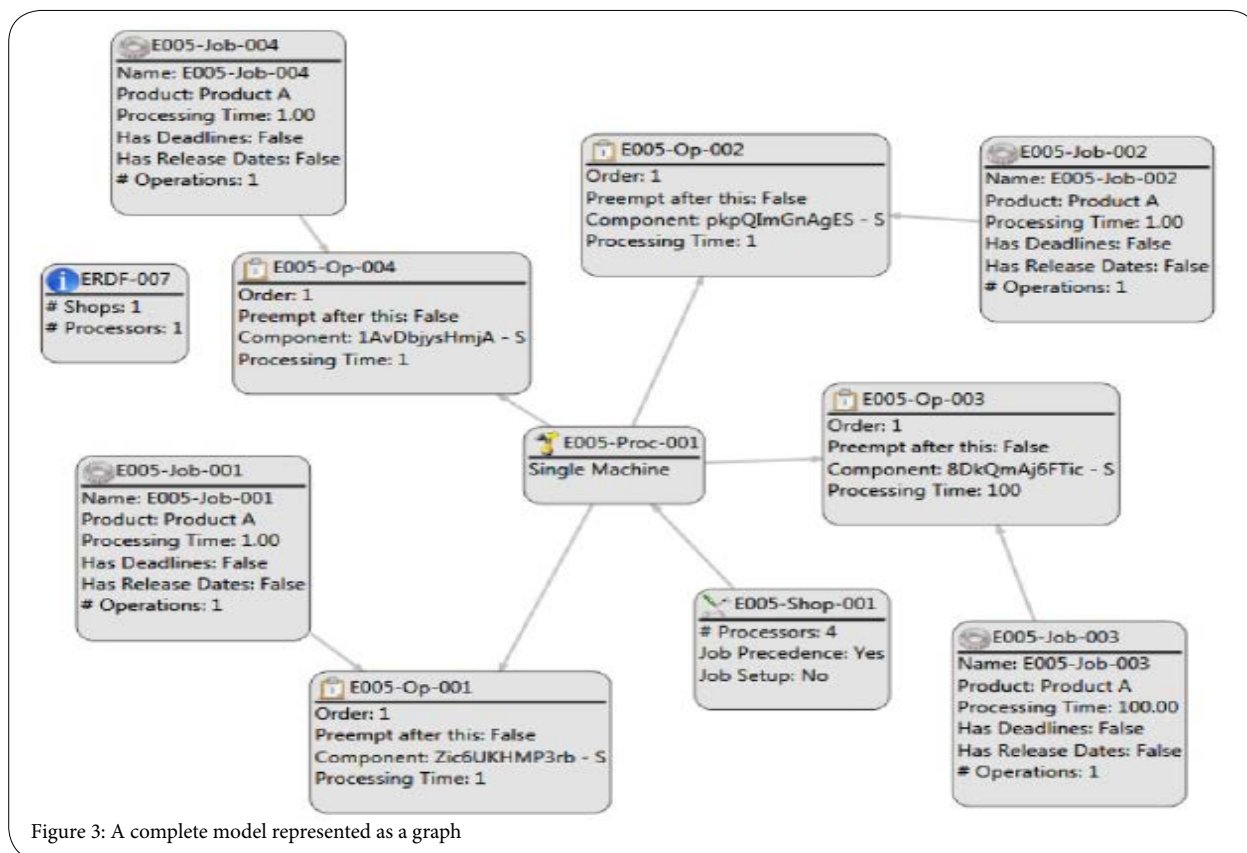


Figure 3: A complete model represented as a graph

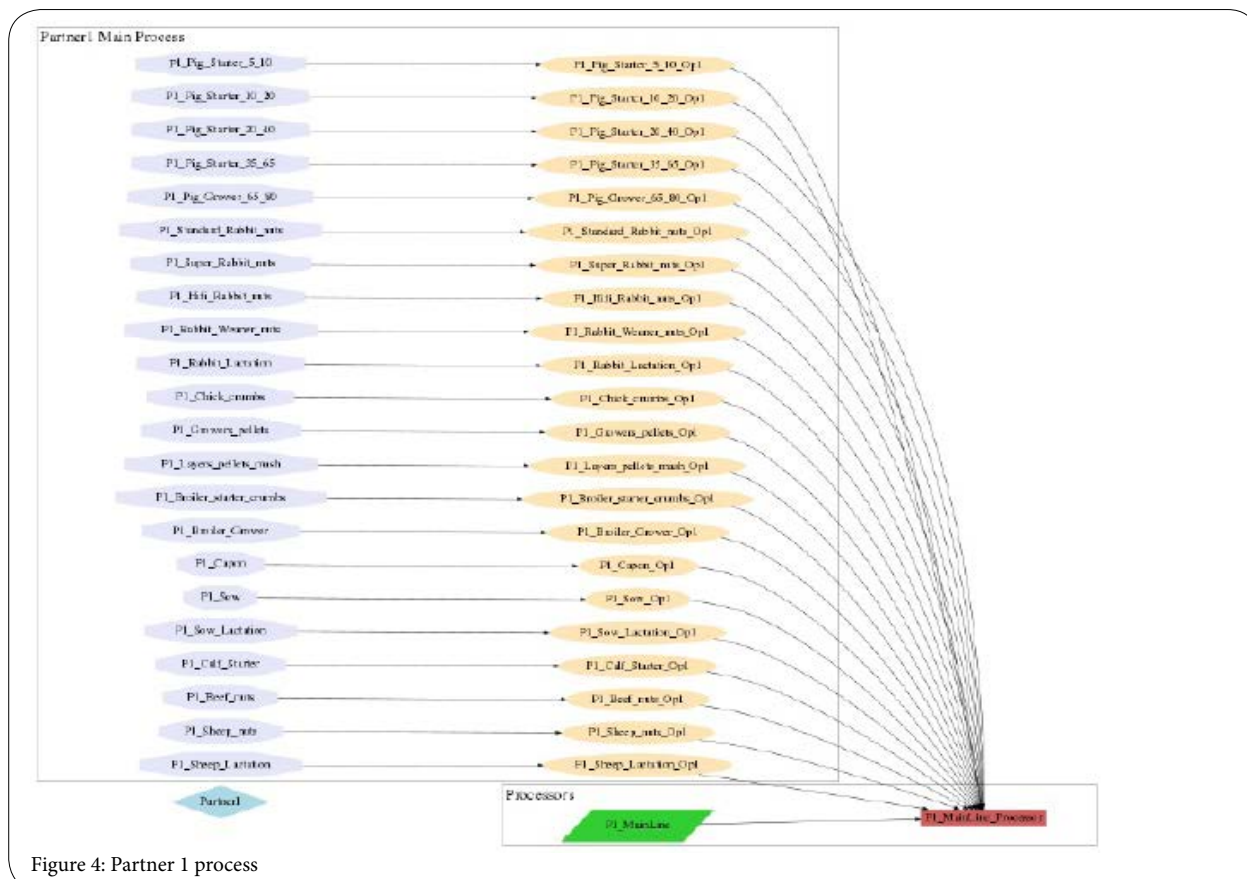


Figure 4: Partner 1 process

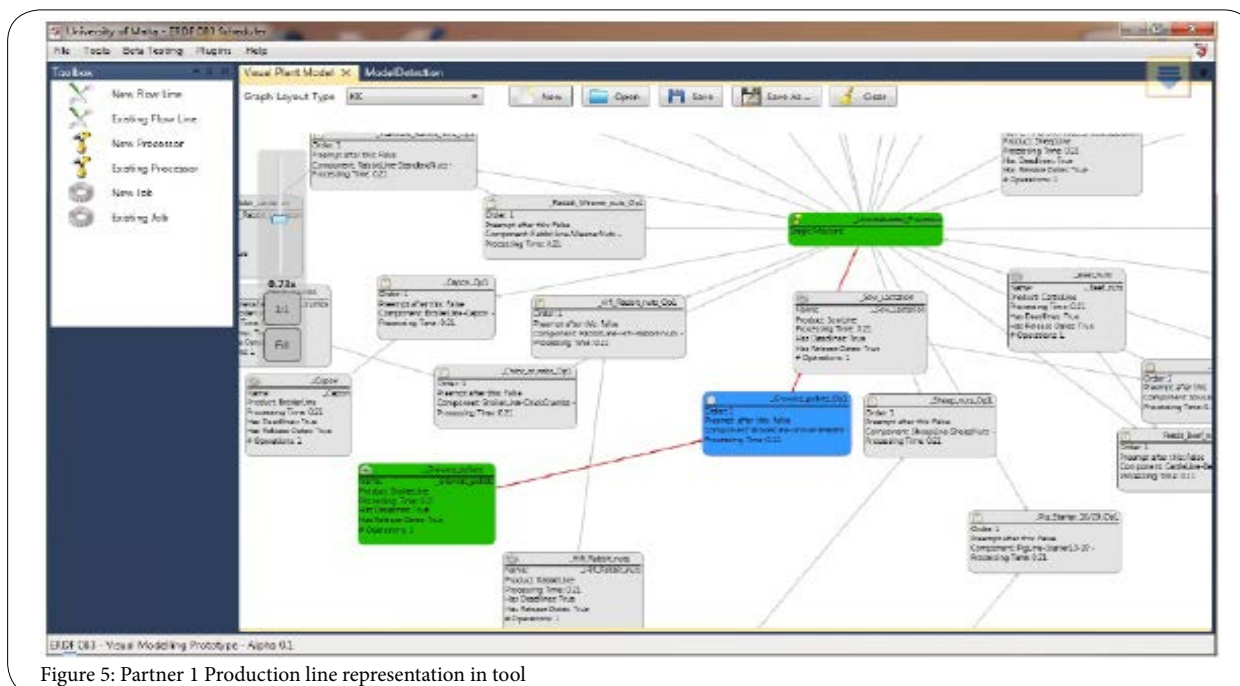


Figure 5: Partner 1 Production line representation in tool

Matching and Related Problems:			
Problem	Problem Type	Reference	Relation
$1 r_{\text{prec}} C_{\text{max}}$	Maximal Polynomially Solvable	Lawler, E.L., 1973. Optimal sequencing of a single machine subject to precedence constraints. Management Sci., 19:544-546.	General Case of Problem

Figure 6: Partner 1 classification result

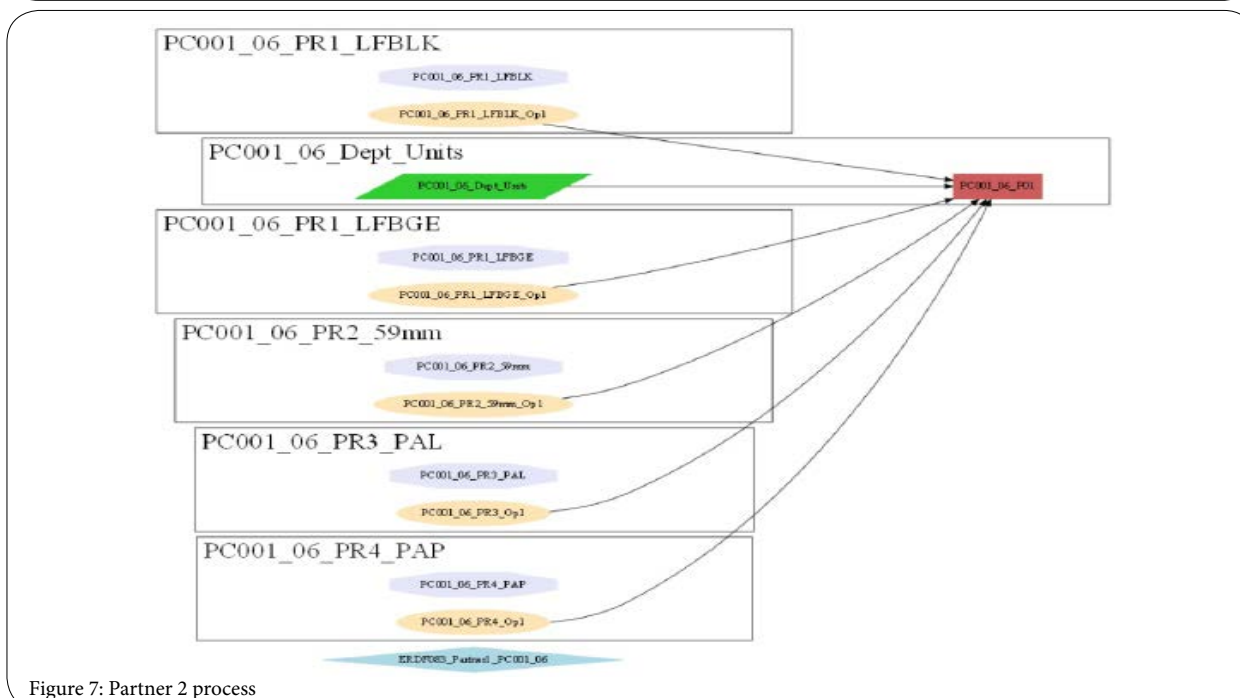


Figure 7: Partner 2 process

Figure 8 shows the model in the modelling tool whereas figure 9 shows the result of the classification. In this case, the tool reported that the problem is maximal NP-Hard and that the problem is solved (i.e.a problem with the same signature already exists in the database indicating a full match).

Use case 3

The use case with the third partner involved the manufacturing of a product where subunits can be manufactured in parallel in different

work centres. While certain subunits can be done in parallel, there is also an element of precedence, where one task is dependent on another to complete before it can be processed.

The individual work centre will represent a flow line in the model. The sequences and sub-sequences in the model are shown in figure 10, and are defined in the tool through a separate process within the tool.

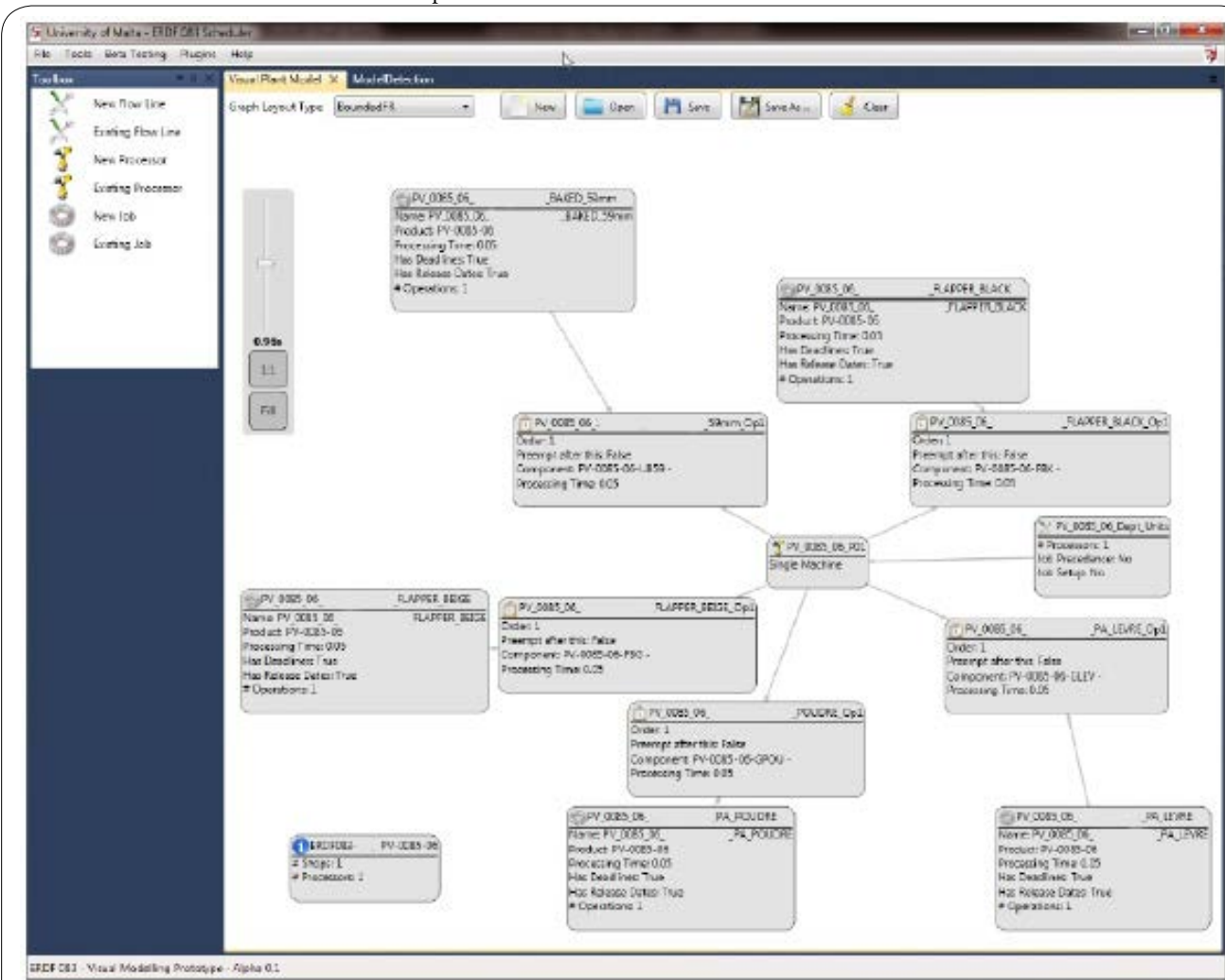


Figure 8: Partner 2 Production line representation in tool

Matching and Related Problems:			
Problem	Problem Type	Reference	Relation
$1 n;d_i;s j C_{max}$	Maximal NP-Hard	Bruno, J., Downey, P. J, 1978. Complexity of Task Sequencing with Deadlines, Set-Up Times and Changeover Costs. SIAM J. Comput., 393-404.	Problem is solved

Figure 9: Partner 2 classification result



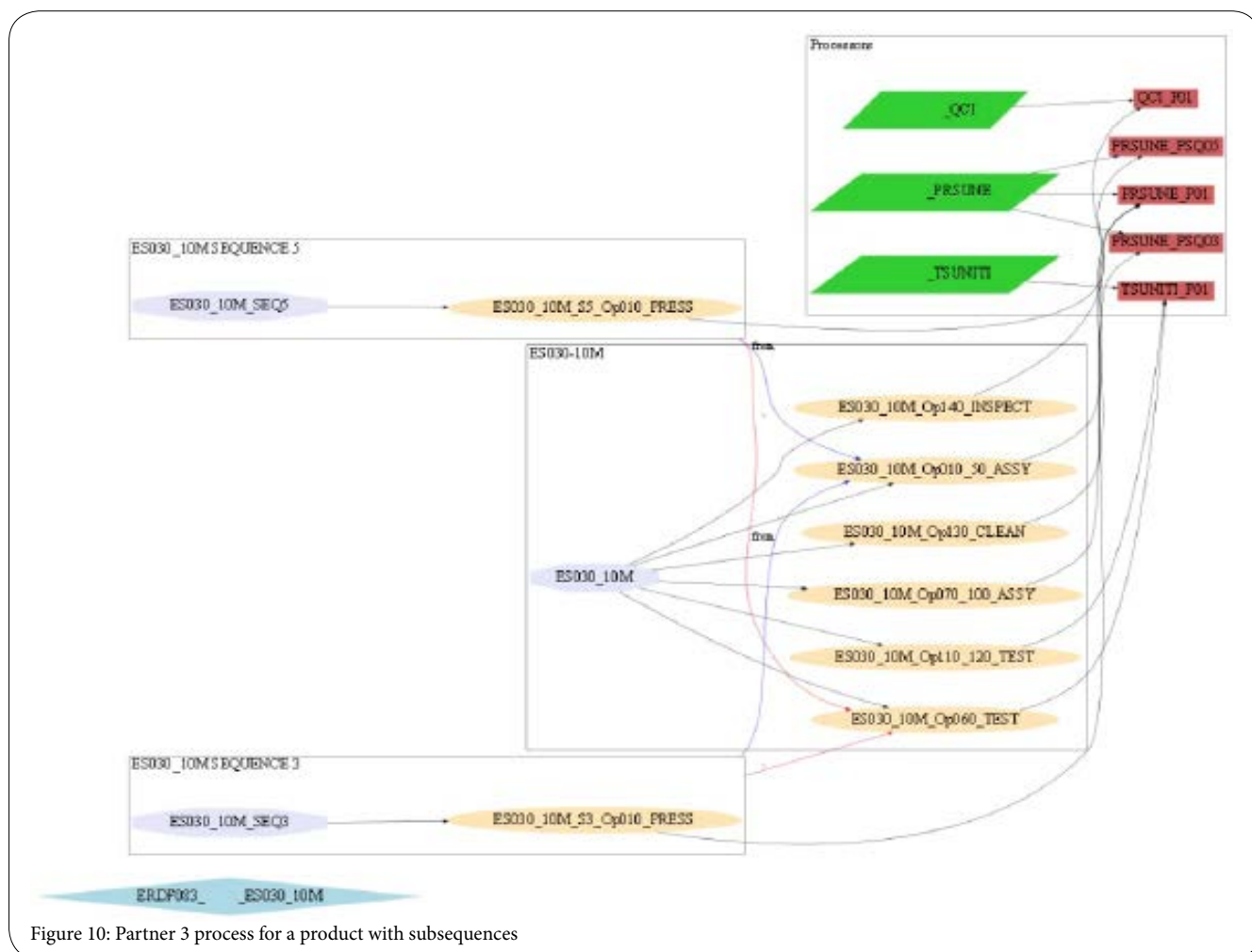


Figure 10: Partner 3 process for a product with subsequences

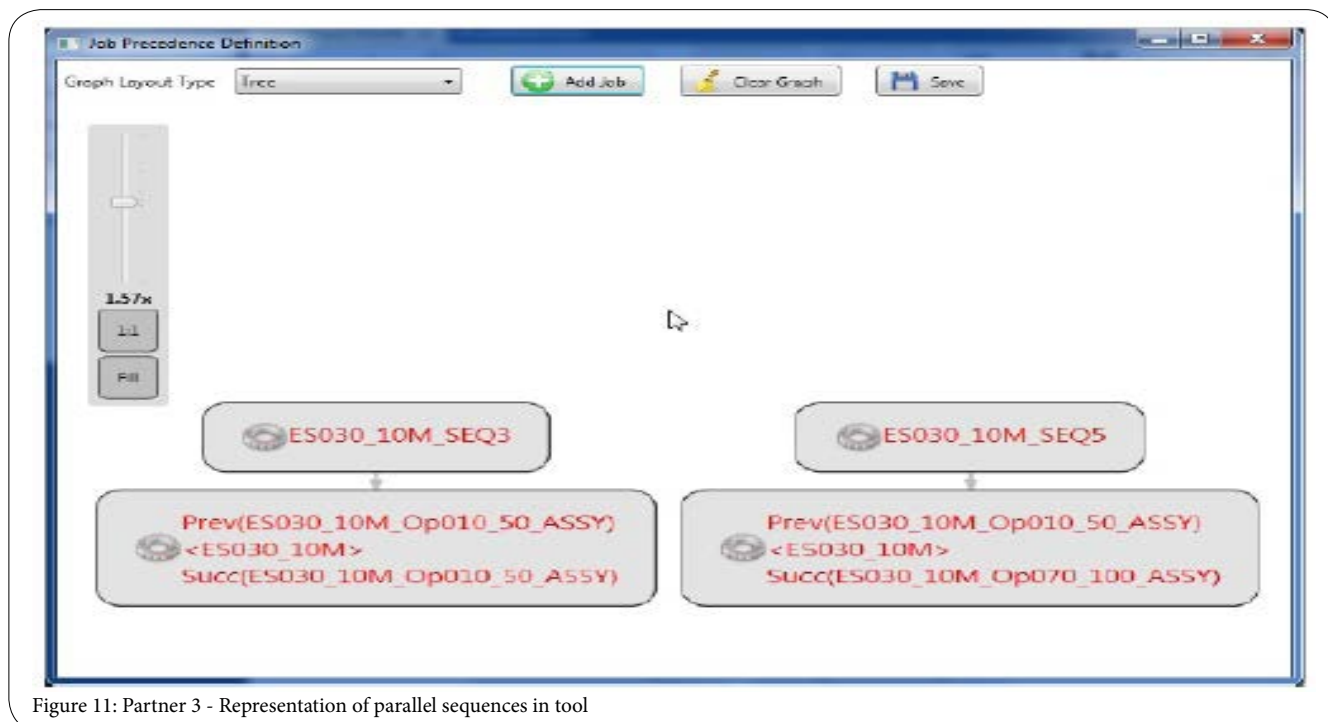


Figure 11: Partner 3 - Representation of parallel sequences in tool

$F|r_0; outtree; d_i | C_{max}$  has complexity Maximal Polynomially Solvable

Matching and Related Problems:			
Problem	Problem Type	Reference	Relation
$F r_0; p_j=1; outtree   C_{max}$	Maximal Polynomially Solvable	Bruno, J., Jones III, J.W., So, K., 1980. Deterministic scheduling with pipelined processors. IEEE Trans. Comput., 29, 4, 308-316.	General Case of Problem
$F r_0; p_j=1; intree   C_{max}$	Maximal NP-Hard	Brucker, P., Knust, S, 1999. Complexity results for single-machine problems with positive finish-start time-lags. Computing., 63, 299-316.	General Case of Problem
$F2 r_1   C_{max}$	Maximal NP-Hard	Brucker, P., Lenstra, J.K., Rinnooy Kan, A.H.G, 1977. Complexity of machine scheduling problems. Ann. of Discrete Math., 1:343-362.	General Case of Problem

Figure 12: Partner 3 classification result - for a subsequence

The classification report generated by the tool in this case will contain a section for each in subsequence detected. In fact, for each subsequence in the model, a signature for the sequence is generated and a classification included in the output of the tool. Figure 12 shows a particular result for a subsequence in the model.

## Conclusion

The present version of the system is capable of modelling simple production lines with simple single machine, classic parallel machine problems and problems involving routing (such as Flow Shops, Job Shops and Open Shops). Additional problem types are currently being added to the database which includes full referencing to the published works.

A future extension to the project will involve extending the engine to handle models with batch processing, limits on machine processing, and buffers for flow shops.

## Conflict of Interest

No authors have a conflict of interest or any financial tie to disclose.

## Acknowledgement

The project is funded in part by the Malta Council for Science and Technology under the European Research and Development Fund.

## References

1. Lenstra JK, Kan AR, Brucker P (1997) Complexity of machine scheduling problems. Annals of discrete mathematics 1: 343-362.
2. M. Pinedo (2009) Planning and Scheduling in Manufacturing and Services. Springer Series in Operations Research and Financial Engineering, Springer Science+Business Media, LLC, 2nd ed.
3. Baker KR, Trietsch D (2009) Principles of Sequencing and Scheduling. Wiley Publishing.
4. M. J. Melitz (2003) The impact of trade on intra-industry reallocations and aggregate industry productivity. Econometrica 71: 1695-1725.
5. M. Edgett (2012) Three hot button areas in improving production efficiency.
6. Blazewicz J, Ecker KH, Pesch E, Schmidt G, Weglarz J, et al. (2007) Handbook on Scheduling: From Theory to Applications. International Handbooks on Information Systems. Springer.
7. Asprova Corporation (2008) Lean Manufacturing.
8. Leung JYT (2004) Handbook of Scheduling: Algorithms, Models and Performance Analysis. Computer and Information Science Series, Chapman and Hall/CRC, 1st ed.
9. Bondy A, Murty USR (2007) Graph Theory (Graduate Texts in Mathematics). Springer, 3rd corrected printing ed.
10. Heller J, Schneiderman B (1972) A graph theoretic model of data structures. SIGIR Forum 7: 36-44.
11. Mirza BJ (2001) Jumping connections: A graph-theoretic model for recommender systems. Master's thesis, Virginia Polytechnic Institute and State University.
12. Caramuta DM, Contiggiani F, Tohmé F (2006) Memory and similarity: a graph-theoretic model for case based decision theory. In XLI Meeting of the Argentina Association of Political Economy, Argentina Association of Political Economy.

13. Sanmartí E, Friedler F, Puigjaner L (1998) Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation. *Computers & Chemical Engineering* 22: S847-S850.
14. Sanmartí E, Friedler F, Puigjaner L, Holczinger T (2002) Combinatorial framework for effective scheduling of multipurpose batch plants. *American Institute of Chemical Engineers (AIChE)* 48: 2557-2570.
15. Friedler F, Fan LT (2012) P-graph - introduction.
16. Pantelides CC (1994) Unified frameworks for the optimal process planning and scheduling. In *Proceedings of the 2nd Conference on the Foundations of Computer Aided Operations* (D. W. T. Rippin, J. C. Hale, and J. F. Davis, eds.), American Institute of Chemical Engineers. CAST Division, CACHE.
17. Grossmann I (2010) STN Interface.
18. Graham R (1980) *Mathematics Today: Twelve Informal Essays*, ch. Combinatorial Scheduling Theory 183-211. Springer-Verlag London Limited; Vintage Pr.
19. R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling- a survey," in *Annals of Discrete Mathematics* (B. H. K. P. L. Hammer, E. L. Johnson, ed.), vol. Discrete Optimization II of 5, pp. 287–326, North-Holland Publishing Company, 1979.
20. Knust S, Brucker P (2009) Complexity results for scheduling problems.
21. Lageweg BJ, Lawler EL, Lenstra JK, Kan AHGR (1981) Computer aided complexity classification of deterministic scheduling problems. Mathematisch Centrum, The Netherlands.
22. Lageweg BJ, Lawler EL, Lenstra JK, Kan AHGR (1982) Computer-aided complexity classification of combinatorial problems. *Communications of the ACM* 25: 817-822.
23. Dürre C (2010) The scheduling zoo.