ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΕΠΙΧΕΙΡΗΣΙΑΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ ΒΑΣΙΣΜΕΝΗ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΠΕΡΙΟΡΙΣΜΩΝ

Διπλωματική Εργασία

του

Δραγοσλή Αθανασίου ΑΜ 16010

Θεσσαλονίκη, 02 / 2019

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΕΠΙΧΕΙΡΗΣΙΑΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ ΒΑΣΙΣΜΕΝΗ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΠΕΡΙΟΡΙΣΜΩΝ

Δραγοσλής Αθανάσιος

Πτυχίο Πληροφορικής, ΕΑΠ, 2014
MSc in Information Systems, ΠΑΜΑΚ, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων καθηγητής

Σακελλαρίου Ηλίας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26/02/2019

Σακελλαρίου Ηλίας	Βεργίδης Κωνσταντίνος	Σιφαλέρας Άγγελος
....	....	....

Δραγοσλής Αθανάσιος

....

Περίληψη

Κατά τις τελευταίες δύο δεκαετίες παρατηρείται μια μεταστροφή στη διαλεκτική της επιστήμης της Διοίκησης Επιχειρήσεων αναφορικά με τον τρόπο οργανωσιακής δόμησης μιας επιχείρησης. Η παραδοσιακή στρατηγική οργάνωσης σε σχεδόν απολύτως διακριτά ιεραρχικά επίπεδα, αντικαθίσταται σταδιακά από τη λειτουργική δόμηση βασισμένη στις επιχειρηματικές διαδικασίες που διατρέχουν οριζόντια και κάθετα την επιχειρηματική οντότητα. Η επιχειρηματική διαδικασία μπορεί να οριστεί ως ένα σύνολο από συντονισμένες ενέργειες που επιδιώκει την επιτυχή υλοποίηση ενός επιχειρηματικού στόχου και την παραγωγή υπεραξίας. Συνεπώς, ο σχεδιασμός και τρόπος εκτέλεσης μιας διαδικασίας μπορεί να επηρεάσει άμεσα την αποδοτικότητα ενός οργανισμού και την ποιότητα που απολαμβάνει ο καταναλωτής από το παραγόμενο προϊόν. Η σημαντικότητα σχεδιασμού και διαχείρισης των επιχειρηματικών διαδικασιών για την επιβίωση μιας επιχείρησης μέσα στο σύγχρονο παγκοσμιοποιημένο περιβάλλον οικονομικής δραστηριότητας, καταδεικνύουν την αναγκαιότητα ανάπτυξης συστημάτων υποστήριξης των αποφάσεων για τη βελτίωση ή αναδόμηση των διαδικασιών με βάση ποσοτικά και ποιοτικά κριτήρια. Λαμβάνοντας υπόψη ότι ο λογικός προγραμματισμός με περιορισμούς μπορεί να προσφέρει το υπολογιστικό περιβάλλον επίλυσης περίπλοκων συνδυαστικών προβλημάτων, στόχος της παρούσας μελέτης είναι ο σχεδιασμός και η ανάπτυξη ενός συστήματος σύνθεσης πολύπλοκων βελτιστοποιημένων διαδικασιών από απλούστερα μοντέλα με βάση προκαθορισμένα κριτήρια αξιολόγησης. Το προτεινόμενο σύστημα αποτελείται από τέσσερα συνεργαζόμενα αρθρώματα λογισμικού. Η βάση δεδομένων αποθηκεύει τα στοιχειώδη μοντέλα επιχειρηματικών διαδικασιών σε μορφή ποσοτικής αναπαράστασης που είναι κατάλληλη για τη διαδικασία βελτιστοποίησης. Η γραφική διεπαφή δίνει τη δυνατότητα στο χρήστη να εισάγει τα απαιτούμενα δεδομένα και δημιουργεί το λογικό πρόγραμμα περιορισμών χρησιμοποιώντας τη γλώσσα λογικού προγραμματισμού Prolog. Η πλατφόρμα λογικού προγραμματισμού ECL^{i}PS^e εκτελεί το λογικό πρόγραμμα με στόχο αφενός να συμπεράνει τις εφικτές διαδικασίες για δεδομένα σύνολα πόρων εισόδου και εξόδου, αφετέρου να κατασκευάσει τις διαδικασίες με την βέλτιστη αξιολόγηση. Τέλος, το υποσύστημα οπτικοποίησης μεταφράζει τη ποσοτική αναπαράσταση των βελτιστοποιημένων επιχειρηματικών διαδικασιών σε γραφική μορφή.

Λέξεις Κλειδιά: επιχειρηματική διαδικασία, λογικός προγραμματισμός περιορισμών, βελτιστοποίηση επιχειρηματικών διαδικασιών.

Abstract

During the last two decades a change occurred in the dialectics of Business Administration science about the principles that should rule the organizational structure of a company. A structural model based on business processes that run through the business entity horizontally and/or vertically, is gradually replacing the traditional strategy that stratifies the functional entity in distinct and isolated hierarchical levels. A business process can be defined as a set of coordinated activities aiming at the successful realization of a business goal and the production of surplus value. Therefore, the design and the implementation of a process can significantly affect the efficiency of an organization's operation and the quality that the consumer receives from the produced outcome. The significance of planning and managing the business processes for the survival of a company in the modern globalized economic environment, demonstrates the need for the development of computational systems that can assist the effort for the improvement or re-engineering of processes in regard to quantitative and qualitative criteria. Considering that the constraint logic programming can offer the computational means for the confrontation of complex combinatorial problems, the aim of this thesis is to design and develop a system that can compose optimized complex business processes from simple tasks based on predefined evaluation criteria. The proposed system is consisted by four collaborating software modules. The database that stores the simple tasks in the appropriate quantitative representation for the optimization procedure. The graphical user interface (GUI) that accepts the required input by the decision maker and creates the constraint logic program using the programming language Prolog. The ECL$_i$PS$_e$ platform executes the logic program to determine the feasible processes for specific input and output resource data sets, and construct the processes with the optimum evaluation. Finally, the visualization subsystem translates the quantitative representation of the optimized business processes into a graphical form.

Keywords: business process**, c**onstraint logic programming, business process optimization.

Contents

Figures

Tables

1 Introduction

1.1 Issue description

During the last two decades a shift in the dialectics of Business Administration science about the principles that should rule the organizational structure of a company has occurred. A structural model based on business processes that run through the business entity horizontally and/or vertically, is gradually replacing the traditional strategy that stratifies the functional entity in distinct and isolated hierarchical levels. This major change in managerial theory and practice enforced by the augmenting need of enterprises to raise productivity, reduce costs, increase production efficiency and develop flexibility, in order to survive and flourish within the current volatile economic environment (Ahmadikatouli and Aboutalebi, 2011; Dumas et al., 2013). According to modern theories of Business Administration, business processes constitute the core asset of an enterprise and the organizational structure must be built upon functional processes rather than administrative hierarchies. Considered as the main structural element, business processes specify the allocation of actions and responsibilities within the company, integrate the available resources with the systems and influence the flexibility of the company to adapt to market changes.

Therefore, the management of business processes can determines the financial profile of the organizations and their potentiality to achieve their objective goals in an economic context (Lohrmann and Reichert, 2013; Dumas et al., 2013; Wibig, 2013). The fundamental role of business processes for achieving competitiveness and profitability, lead managers to focus on methodologies that can assist their continuous improvement and optimization, aiming to establish a solid competitive advantage and improve the overall performance (Vergidis, Tiwari and Majeed, 2006b). Consequently, the administration of business processes has become a major concern and the concept of Business Process Management (BPM) has received wide acceptance (Lohrmann and Reichert, 2013; van der Aalst, 2013). BPM combines knowledge from the scientific areas of business administration and computer science (Weske, 2012; van der Aalst, 2004), in order to develop methods, and tools to support the design, enactment, management, analysis and improvement of operational business processes (van der Aalst, ter Hofstede and Weske, 2003; van der Aalst, 2013; Vom Brocke and Rosemann, 2010).

The major objective of BPM is the business processes optimization, which can be described as the procedure of identifying, analyzing, restructuring and monitoring the business processes in order to maximize the delivered surplus value for the organization and the customer. In this context, the crucial decision about process optimization is the definition of the processes performance metrics such as cost, revenue, duration, execution quality and adaptability to deviations (Dumas et al., 2013). Using the techniques of BPM, managers try to ensure the functional consistency, take advantage of the market opportunities and augment the produced value for the organization and its customers (Dumas et al., 2013; Tiwari, Vergidis and Turner, 2010; Georgoulakos et al., 2017). Therefore, within the context of BPM the computational systems could be valuable tools for the identification and solution of the unstructured or semi-structured problem of optimized business process construction.

1.2 Aim and objectives

Aim of the present thesis is the design and the development of a system that can compose simple business tasks in order to construct optimized complex business processes in the base of predefined quantitative and qualitative evaluation criteria. The constructed optimized processes can facilitate significantly the decision-making activities about processes improvement or reengineering and additionally, can serve as simulation tools where managers can test with safety a plethora of different combinations of tasks or parameters. For the achievement of this aim, we identify four objectives that must be investigated for the confrontation of the following challenges:

1. The classification of the business process optimization problem in a representative class of mathematical problems.
2. The development of an appropriate quantitative representation of both simple tasks and complex business processes.
3. The definition of the qualitative and quantitative evaluation criteria.
4. The formulation of the objective function that incorporates the evaluation criteria and is used as metric for the optimization procedure.

1.3 Problem formulation and contribution

The problem of multi-objective business process optimization is a combinatorial optimization problem that is characterized by high computational complexity due to the non-linear, non-convex and often discontinuous nature of the involved mathematical models (Wang, Salhi and Fraga, 2004). In the context of this thesis, is formed as an extension of the well-known in the operations research domain Resource Constrained

Project Scheduling Problem (RCPSP). The RCPSP presupposes a finite set of tasks with known duration and resources demands, both in terms of type and quantity, along with a set of constraints that describe precedence relationships between the tasks (Bukata, Sucha and Hanzalek, 2015). The objective consists of finding the best feasible solution according to one of more criteria, satisfying precedence relations and the resource availability (Artigues, 2008). In order to extend the RCPSP we consider that the set of tasks which consist the business process is unknown and there is not any information about their precedence relationships. More specifically, the initial available information is a library of tasks $P_{library}$, a library of resources $R_{library}$, a set of input resources $I_{BP} \subseteq R_{library}$ and a set of output resources $O_{BP} \subseteq R_{library}$. The primary goal is to discover if there are feasible process designs $FD \subseteq P_{library}$ that consume all the resources in I_{BP} and produce all the resources in O_{BP}. The tasks of the feasible designs are related with precedence relations in terms of resources' availability and mutual exclusion and can be evaluated by a predefined objective function. The ultimate goal is to determine the best one among the feasible designs.

In more detail, each task consumes resources of specific type and quantity as input and produces resources of specific type and quantity as output. The precedence relations are defined at runtime in the base of input/output resources and additionally, the solution of the optimization procedure has to satisfy a set of predefined mutual-exclusion relations between some of the tasks. A vector of five measurable attributes, namely cost, duration, revenue, quality and flexibility, characterizes quantitatively every task. The attributes of the tasks that are included to a feasible process design are mapped to its corresponding attributes via aggregation functions. The linear combination of the five aggregation functions constitute the multi-objective function to be maximized for the determination of the optimized design. The objective is twofold, firstly to discover feasible designs of a business process that satisfy the precedence and mutual exclusion relations for a certain set of initially input/output resources. Secondly, to decide which of the feasible design the best one using as metric the predefined objective function. The aforementioned formulation of the problem is the foundation for the rationale of the proposed system. In more detail, the framework is consisted by three collaborative modules that are implemented by different software technologies. The first is a relational database, which includes the simple tasks and the available resources. The second is built in the object-oriented language Java and acts as a middleware between the database and the ECL^{i}PS^e platform, which implements the rationale part of the optimization procedure. Additionally, the Java module visualize the optimized business process schedule into a directed tree graph.

1.4 Thesis structure

The rest of the thesis is organized in six chapters. The second chapter is a review of the literature about the research related to business process optimization. Specifically, it provides the reader with the basic terminology and the dominant definitions of the business administration domain and exhibits the most significant scientific approaches that have been developed for the confrontation of the process optimization problem. The third chapter presents comprehensively the main principles of the mathematical paradigms where is based the rationale of the developed optimization procedure. The fourth chapter analyzes in detail the methodology that was used, describing the theoretical formulation of the problem under examination and the construction procedure of the proposed framework. The fifth chapter describes the design and development of a software application that produces instances of the formulated problem. This intervention was deemed necessary for the creation of the experimental data that was used to evaluate the effectiveness of the proposed framework. The sixth chapter initially presents a complete step-by-step execution of the developed process optimization software. Then it exhibits a comparative assessment of the framework with problem's instances of various sizes, which investigates its potential and its limitations. Finally, the seventh chapter summarizes the main conclusions, discusses the limitations and identified some possibilities for the further improvement of the proposed framework.

2 Literature review

In this chapter, we attempt to outline a general insight of the scientific context where is included the research issue of the present thesis. Therefore, the first section presents the semantics of the terms that are used throughout the thesis and explains the definitions that have been derived from the relevant literature and adopted by the scientific community as the most consistent and prevalent. We consider that the comprehensive explanation of the terminology is of major importance for avoiding confusion, due to the plethora of different and often contradictory definitions of the same term that is found in bibliography, especially of the Business Administration domain. The second section presents in detail the main methods and techniques that have been developed from the global scientific community, with respect to the problem of processes optimization within the domain of business administration. Finally, the third section identifies the scientific gap into the arsenal of methodologies that address the problem of business process optimization and states the challenges that must be met by the proposed framework.

2.1 Business Process Management terminology

The modern administrative approaches consider the business process as the fundamental structural element of an enterprise. This principal let to the birth of the Business Process Management scientific domain, in an effort to be developed strategies and methods for the optimization of business processes. This section provides a comprehensive description of the aforementioned terms.

2.1.1 Business Process

There is not a generally accepted definition of business process in the relevant literature. Different authors have suggested a variety of definitions according to their personal research perspective, focusing on a subset of the aspects that constitute the notion of business process (Ahmadikatouli and Aboutalebi, 2011;Volkner and Werners, 2000). Harvey (2005) specifies the process as a "*step by step rules specific to the resolution of a business problem*". Hammer and Champy (1993) emphasize on the input-output functionality, describing process as a collection of activities that consumes an input and produces an output with surplus value for the customer. Davenport (1993) introduces execution constraints between activities, defining a business process as "*a set* *of tasks logically related in a specific order across time and place with a beginning, an end, and predefined inputs and outputs. The execution of those tasks aims to achievement of a business outcome for a particular customer or market*". Vergidis and Tiwari (2008) perceive the notion of business process as a "*collective set of tasks that when properly connected perform a business operation that is producing value to the organization*". Doumas et al. (2013) define the business process as a "*collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer*". Similar definitions can be found in Volkner and Werners (2000), Weske (2012), Tiwari, Vergidis and Turner (2010) and Melao and Pidd (2000).

In general, a business process consists of an arbitrary number of events, activities, activity attributes, input/output resources, actors and connectivity patterns. A very simple activity is called a task. Events are intangible entities that happen atomically and may trigger the execution of an activity. The attributes are measurable characteristics of the activity such as cost, revenue, duration, execution quality and flexibility that can be used for the evaluation of a business process. A process involves a finite number of actors, which can be humans, software systems, physical objects like equipment, tools and products or immaterial objects like electronic documents and records. The patterns that interconnect the activities constitute one of the main distinguishing characteristics of the business processes. Finally, the execution of a business process produces an outcome, which is consumed by a special actor that is called customer (Tiwari, Vergidis and Turner, 2010; Dumas et al., 2013).

2.1.2 Business Process Management (BPM)

Business Process Management is defined as an administrative strategy that is consisted of a variety of concepts, methods, and techniques to support the identification, design, analysis, monitoring, reengineering and enactment of business processes (Weske, 2012;Dumas et al., 2013; Van der Aalst, ter Hofstede and Weske, 2003). The above definition declares the sovereign role of business process in the context of BPM and the fact that BPM is a circular procedure that comprised of six phases (fig. 1.1) (Dumas et al., 2013):

1. Process identification: this phase addresses a business problem and analyzes the relevant to that problem functions and dependencies. The outcome is an updated

architecture that provides an overall view of the processes and their interconnections.

2. Process discovery or as-is modeling: the result of this phase is an as-is model which documents the current state of the process. For this purpose many alternative modeling notations have been employed, either specific in the task or more general, like Business Process Modeling Notation (BPMN), Petri nets, Event-Driven Process Chain (EPC), Unified Modeling Language (UML), Business Process Execution Language (BPEL) and so forth (Weske, 2012; van der Aalst, 2013).
3. Process analysis: all the issues that are relevant to as-is processes are identified, analyzed, documented and evaluated by qualitative performance measures. The output is a structured hierarchical ranking of issues.
4. Process redesign or improvement: the goal of this phase is to discover changes to the current state of processes that will lead to their improvement and will help the organization to achieve its business objectives. The output is a to-be process model, which serves as a guide for the next actions.
5. Process implementation: during this phase the required changes are prepared and applied in order to move from the as-is to the to-be process.
6. Process monitoring and controlling: the redesigning process is supervised and the collected data determines if and how well the process complies with the predefined performance objectives. Problematic deviations and unexpected behaviors lead to corrective actions and the cycle of BPM is repeated on a continuous basis.

The most important gain of BPM is the introspection of the operation of an organization. Through BPM an enterprise can create optimized processes that operate consistently at the level of which they are capable. The operational benefits of consistency in terms of cost, speed, quality, and service, can be transformed into a higher customer loyalty and a strong competitive advantage for the organization (Hammer, 2015).

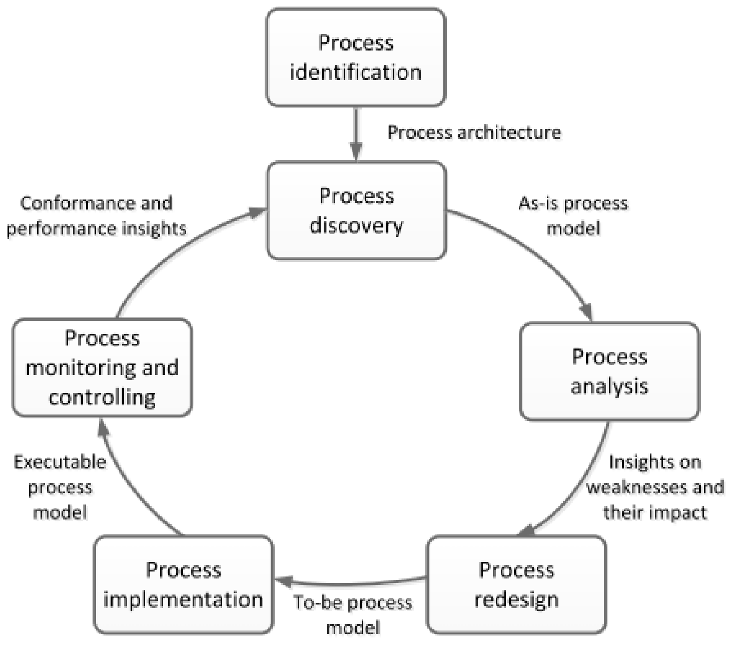


Figure 2.1.BPM life cycle (Dumas et al., 2013, p.21)

2.1.3 Business Process Optimization (BPO)

The plethora of existing definitions for the term "business process" is a clear evidence for the vagueness of its nature and the difficulty of constructing a mathematical representation for an optimization procedure. According to Wang, Salhi and Fraga (2004), the non-linear, non-convex and discontinuous nature of the involved mathematical models, make the implementation of process optimization a difficult task. Formally, BPO can be defined as the problem of constructing feasible business process schedules with optimum values for execution time, cost, outcome quality and customer satisfaction (Ahmadikatouli and Aboutalebi, 2011;Georgoulakos et al., 2017;Tiwari, Vergidis and Turner, 2010). Although the concept of optimality is fuzzy and there is not a broadly accepted methodology for business process optimization (Wibig, 2013;Vergidis and Tiwary, 2008;Vergidis, Tiwari and Majeed, 2007), an abstract view of BPO procedure includes the following steps:

1. Data collection from many different sources and integration in a consistent and homogeneous structure.
2. Comprehensive data analysis with statistical and data mining techniques, for the evaluation of predefined metrics and the discovery of hidden insights and bottlenecks.
3. Detection and application of improvements for the confrontation of the revealed deficiencies (Wibig, 2013;Niedermann and Schwarz, 2011).

2.2 Business process optimization approaches

In general, the research related to business process optimization can be distinguished into three categories: the general management techniques, the BPM frameworks and the algorithmic approaches (Lohrmann and Reichert, 2013).

2.2.1 General management techniques

This category includes managerial methods that aim mainly to the evaluation of the overall performance of business processes. Afterwards, the obtained results can be used for decision making about the improvement initiatives. However, they do not support any automated procedure and the plan of process optimization is left to stakeholder's intuition (Lohrmann and Reichert, 2013). Two widely adopted techniques of this category are:

1. **Benchmarking**. It is a technique for the improvement of business processes of an organization by the combination of experience and knowledge acquired from the comparison to other organizations that are recognized as the best within the relevant business context (Bhutta and Huq, 1999). The philosophy of benchmarking is based on the identification of the highest implementation standards of comparable processes in leading competitors and the collection of the necessary information that can be utilized in a business processes improvement procedure (Bhutta and Huq, 1999).

   Two are the key factors for a successful application of benchmarking and the effective exploitation of its results: the selection of the appropriate performance measures that can be mapped directly to the business objectives of the company and the proper adaption of the selected "best practice" to the specific structural characteristics of the domestic organization (Bhutta and Huq, 1999). Benchmarking can be distinguished in qualitative, which compares the as-is state of a process with known good practices and is mainly related to evaluation of organizational structures, business processes models and information systems. Quantitative benchmarking utilizes predefined key performance indicators in order to measure various characteristics of a business process and compare the results with those from similar competitive organizations (Lohrmann and Reichert, 2013).
2. **Business Balanced Scorecard**. It is a powerful tool for the evaluation of the business processes performance, which can give a comprehensive view of the entire company (Kaplan and Norton, 1992). Balanced scorecard is consisted by a

set of measures that consider the organization through four dimensions, namely, the “Financial”, the “Customer, the “Innovation and Learning” and the “Internal Business”, and its advantage is that minimizes the information overload by limiting the number of utilized measures (Kaplan and Norton, 1992). The main characteristic of scorecard is that includes in a single management report, a combinatorial overview of the performance of heterogeneous aspects of a company's operational functions. Additionally, the scorecard tries to avoid sub-optimization by forcing managers to a holistic consideration of all of the operational measures concurrently (Kaplan and Norton, 1992). However, the drawback of scorecard is that in regard to each organization requires the development of special tangible and intangible key performance indicators for each dimension (Lohrmann and Reichert, 2013).

2.2.2 BPM Frameworks

Frameworks are not considered as methodologies or managerial techniques but rather as holistic strategies that introduce a new “way of thinking” about structuring and administering an organization. Just as all the other BPM strategies, frameworks aim to the functional improvement of business processes in order to augment the overall efficacy of the enterprise. However, although they offer a rich arsenal of managerial guidelines and tools, in practice there is a rather wide margin of interpretation and the initiative is based upon manager’s experience. Next, follows a discussion about some of the most popular frameworks in BPM literature.

2.2.2.1 Business process reengineering (BPR)

BPR is a revolutionary management strategy focusing on both how an organization is structured and how it can be improved. According to BPR, a company should not be seen in terms of functions, departments or products but rather as a set of interconnected key business processes that interact for the achievement of predefined strategic goals. The optimization of these processes can be conducted by their ground-up redesigning, rather than the partial improvement of their constructing components (Davenport, 1993). BPR aims to assist organizations to recreate their structure through the radical redesign of their processes, employing whatever administrative innovations, new technologies and available business resources.

This combination of the process-oriented point of view of the organization with the application of innovations to key processes can create tremendous potential for the achievement of business objectives such as cost and time reduction, quality improvement and operational flexibility (Davenport, 1993). However, the radical nature of BPR strategy can emerge serious danger even for the existence of a company, due to the lack of a well-defined procedure and a comprehensive toolkit of techniques that can facilitate its practical applicability and ensure the completeness of coverage of the business objectives (Lohrmann and Reichert, 2013; O'Neill and Sohal, 1999). Indeed, relevant studies in US and Europe have shown that among the applied BPR initiatives only 30 percent can be considered successful because of misuse in implementation (O'Neill and Sohal, 1999).

2.2.2.2 Lean manufacturing

Lean manufacturing is a customer-oriented production practice that aims to the elimination of the wasteful components of business processes. This approach considers as waste any consumption of resources or any function that target to any goal other than the creation of value for the end customer. More specifically, any business process may contain mistakes that require rectification, actions which are not actually needed, transportation of goods or humans without any purpose, waiting queues that waste time and the production of items which do not meet the needs of the customer (Womack and Jones, 2003). Lean manufacturing provides a way to do more with less, specifying value, sequencing the value creating actions in the optimized way, conducting the actions without interruption whenever someone requests them and making work more satisfactory by providing immediate feedback on efforts to convert waste into value.

The pursued goals of Lead manufacturing are the maximum efficiency, the clearly defined responsibilities, the exact description of processes and the traceable communication ways (Womack and Jones, 2003). *Lean management* is an extension of Lean manufacturing and in the core of its philosophy is the principle that the customers are willing to pay for value, not for mistakes or waste. Therefore, in order to maximize profit an organization has to increase the value of produced goods or services (Womack and Jones, 2003). In general, Lean manufacturing is based on five main principles (Womack and Jones, 2003):

1. *Specify Value*. The critical starting point for Lean manufacturing is the precise definition of value in terms of a specific product that is offered at a specific time and price and has certain characteristics defined by a specific customer.
2. *Identify the Value Stream*. The value stream is the set of actions required to deliver a product from the concept to the hands of the customer. The identification of the entire value stream for each product can eliminate a significant amount of waste.
3. *Production Flow*. This principle pursues the construction of a continuous execution flow of the value creating actions. According to Lean manufacturing, higher efficiency and accuracy can be accomplished when a product is manufactured by a continuous production line from raw material to finished good. Thus, the focus must be on the product and its specifications rather than the organization or the equipment.
4. *Demand Pull*. It is the ability to design and schedule exactly what the customer wants just when the customer wants it. Therefore, a company must let the customer pull the product as needed rather than trying to push the products to the customer.
5. *Perfection*. The implementation of the four initial principals involves an interaction with each other in an endless execution cycle. Although the effort for the improvement of business processes in terms of time, cost and mistakes reduction can never stop, Lean manufacturing offers the organizational means for the convergence towards the perfection.

2.2.2.3 Total Quality Management (TQM)

TQM is a holistic and structured administrative strategy that aims to integrate all processes within an organization into a procedure for the continuous improvement of competitiveness, effectiveness and flexibility (Oaklant, 2003). This goal can be achieved through the ongoing structural and functional adjustment of business processes in response to the received feedback from all possible sources (Godfrey, 1999). TQM is essentially a way of planning, organizing and understanding each activity by each individual at each level. To attain the promotion of business efficacy, the principles of TQM must be propagated throughout the organization, along with participants' commitment to accept the burden of realization's responsibility. On the other hand, the organization needs to ensure participants that through the efforts and achievements they will acquire the analogous recognition and reward (Oaklant, 2003). Although the plethora of different approaches, in general, a TQM model that covers all angles and aspects of an organization is based on the following concepts (Oaklant, 2003; Godfrey, 1999; Omachonu and Ross, 2004):

1. *Planning*. TQM requires the establishment of a strong top-management leadership for the development and deployment of clear short, mid and long-term vision, policies and strategies that utilize properly its concepts and scientific methods. The implementation of TQM can be conducted by defining the mission, identifying the strategic goals, discovering customer's requirements and determining the activities required to fulfill these objectives.
2. *Performance*. Regarding vital organizational resources such as infrastructure, humans and information, a performance measure framework with the appropriate metrics must be developed for carrying out self-assessment, audits, reviews and benchmarking.
3. *Processes*. In the core of TQM is the notion of business process. Therefore, first priority is the introduction of a cross-functional quality management system with the potential to understand, analyze, design and redesign the business processes, combining all the fundamental organizational powers in order to achieve continuous improvement.
4. *People*. TQM is not a strict framework with specific guidelines about human participation within an organization. It is rather a way of thinking that pursues a cultural change that accepts as primary objective to fulfill customer requirements and implements a managerial philosophy that acknowledges the ethic of continuous improvement. This cultural change can be achieved by the encouragement of employees' involvement through the constant training and education, the teamwork, the creation of effective communication channels, the diffusion of information and feedback throughout the organization, the sponsoring of innovation and excellence and the formation of a supporting working environment. The objective is to make individuals accountable for their own performance and to remain committed in attaining the quality targets of organization.

2.2.2.4 Six Sigma

Six Sigma is a statistical methodology closely related to the TQM and Lean Management. The goal of Six Sigma is the continuous improvement of business processes by reducing the causes of defects and the elimination of the variability in products manufacturing through the removal of normally distributed errors. Six Sigma utilize a variety of techniques to analyze comprehensively a business process, aiming to the discovery of possible problematic element and the reformation of unnecessary or inefficient steps that reduce the overall performance level of the process (Conger, S., 2010). In the context of Six Sigma can be distinguished two project life cycles, DMAIC that stands for Define-Measure-Analyze-Improve-Control and DMADV that is translated to Define-Measure-Analyze-Design-Verify.

In general, the DMAIC approach is recommended for the improvement of an existing business process and the DMADV approach is suitable for the design of a new process (Conger, S., 2010). However, the application of Six Sigma in practice suffers by a major drawback. Although it involves a large set of techniques for the recognition, analysis, evaluation and redesign of a business process, it does not offer specific guidance about which methods are the best in a certain phase or state of the process aspects under consideration. Additionally, there is little concern about how to customize or improvise the applied techniques in regard to specific cases and their variations (Conger, S., 2010).

2.2.2.5 Quality of Business Processes (QoBP)

The suggested by Heravizadeh, Mendling and Rosemann (2009) framework, defines business process quality in terms of 41 quality dimensions divided in four categories: quality of functions, quality of input and output objects, quality of non-human resources, and quality of human resources. The quality dimensions guide the process analysis that conducted by the proposed Process Root Cause Analysis technique (PRCA). This approach is founded on the assumption that an issue can only be solved by addressing the underlying cause and pursues to identify the consisting elements and the relations of a process for a particular case (Heravizadeh, Mendling and Rosemann, 2009). In particular, the PRCA process is consisted by six major steps for the definition of a business process model, a quality model for a process, a goal model, correlation model, measurement model for each goal and the identification of the issue occurrences.

The crucial step is the availability of a generic quality model that directs the creation of the goal model and the precision of the subsequent steps. The objective of a quality model is to identify all the potential quality requirements for the elements of the process (Heravizadeh, Mendling and Rosemann, 2009). QoBP framework combines goal-oriented and activity-oriented process modeling for the identification of a process's weaknesses and an explicit description of its quality attributes (Heravizadeh, Mendling and Rosemann, 2009). However, the suggested approach does not show the quality dimensions interrelation to organizational targets or to an overall formal quality definition (Lohrmann and Reichert, 2013). Additionally, PRCA approach does not define clearly the completeness the quality dimensions and a specific procedure for the evaluation of the overall process quality (Lohrmann and Reichert, 2013).

2.2.3 Algorithmic approaches

A considerable number of algorithmic approaches have been developed in order to deal with individual performance aspects of business processes such as execution quality, control flow and resource scheduling. Although they do not comprehensively cover the entire field of business process optimization, they constitute an exceptional set of alternative tools in the context of BPM (Lohrmann and Reichert, 2013).

2.2.3.1 Graphical approaches

A class of research efforts tries to achieve optimization using as main tool the business process graphical models. Van der Aalst (1998) proposed a framework that uses Petri nets as the medium for the achievement of business process optimization. Petri nets are directed bipartite graph with two type of nodes, places and transitions, which can be connected via directed arcs with the constraint that connections between nodes of the same type is not allowed (Van der Aalst, 1998). The advantages of Petri nets are their well-defined formal semantics and the identification of a plethora of mathematically founded properties. In the base of these characteristics have been developed many powerful analysis techniques for Petri nets such as linear algebraic techniques for the verification of properties, coverability graph analysis, model checking, and reduction techniques that analyze their dynamic behavior, simulation and Markov-chain analysis for the model performance evaluation (Van der Aalst, 1998).

Van der Aalst used the high-level Petri nets, an extension of classical Petri nets with color, time and hierarchy in order to represent specific aspects. In particular, the color extension models the value for an attribute of an entity of the graph. The extension with time describes the temporal behavior of the system and the extension with hierarchy confronts specification's complexity using sub-nets for the construction of the entire model (Van der Aalst, 1998). In order to achieve business processes optimization, Van der Aalst (1998) defined formally the notion of the soundness of workflow nets as the set of minimal requirements that any workflow process definition should satisfy. Toward the same direction, Hallerbach, Bauer and Reichert (2009) tried to expand the notion of workflow soundness for an entire process family. More specifically, for a particular business process there can be in practice a significant number of variations due to the special requirements of the business context and the set of constraints that must be satisfied. In their work Hallerbach, Bauer and Reichert (2009), examined advanced concepts and developed a five-step effective algorithm that utilizes these concepts, in the context-based and constraint-based configuration of variations that belong to the same process family while guaranteeing their property of soundness.

Going further, Reichert, Rinderle and Dadam (2009) proposed an extension of workflow soundness in the context of dynamic processes that change at run-time. Motivated by the fact that a static way of thinking can lead to the design of business processes that are rigid and cannot be reined any further, they proposed the ADEPT2 framework as a Process-Aware Information System, which allows the dynamic structural adaptation and evolution of process models in order to confront uncertainty, functional exceptions and context variations during execution. ADEPT2 pursues to achieve the ability of processes to deal with uncertainty, the production of new process instances through structural adaptation and the evolution of processes in order to comply with emerging specification. The ultimate goal is to preserve the correctness of the modified process instances guaranteeing their structural and behavioral soundness and to ensure that the model adaptations do not violate at run-time the constraints that was set at build time (Reichert, Rinderle and Dadam, 2009).

The evolution of Process-Aware Information Systems (PAIS) gives the ability to configure a business process at design-time and adapt it dynamically at runtime (Li, Reichert and Wombacher, 2010). However, although PAIS offer run-time flexibility while preserving model’s consistency and lead to the construction of a large number of variants from the same model, in general it is a computationally expensive and difficult to maintain procedure (Li, Reichert and Wombacher, 2010). For the containment of this drawback, Li, Reichert and Wombacher (2010) developed the MinAdept project that utilizes an efficient clustering algorithm for knowledge mining from a collection of process variants and the construction of a model with minimized average distances from the variant models. Subsequently, the constructed model can be used as reference model by the PAIS in order to decrease the complexity during processes configuration and adaptation.

2.2.3.2 Evolutionary techniques

However, the high complexity of some real-world multi-objective optimization problems is challenging the computational adequacy of traditional operational research techniques (Coello, Lamont and Van Veldhuizen, 2007). Wang, Salhi and Fraga (2004) characterize process optimization as a difficult task because of the non-linear, non-convex and often discontinuous nature of the involved mathematical models. This fact turned the focus of researchers to the evolutionary computation due to the population-based nature of evolutionary algorithms that allows the generation of several elements of the Pareto optimal set in a single run (Coello, Lamont and Van Veldhuizen, 2007). In the context of business process optimization, the evolutionary or genetic algorithms can act on a large population of process designs. Therefore, they evaluate many alternative models generating a series of diverse process designs on the basis of specific objectives and they can discover designs that a human designer may ignore (Tiwari, Vergidis and Turner, 2010). According to Moon and Seo (2005), the most attractive characteristic of evolutionary algorithms is their flexibility to solve a variety of objective functions with the least mathematical requirements.

Wang, Salhi and Fraga (2004) proposed an evolutionary approach for business process optimization, which extends their pattern-matching Scan Circle genetic algorithm with the Learning Vector Quantization (Kohonen, 1995 cited in Wang, Salhi and Fraga, 2004, p.663). Their approach defines the crossover and mutation operators on the basis of the extracted knowledge about the feasible region and the behavior of the objective function during the data analysis, aiming to the discovery of the key aspects of the search space the objective function. The combination of visualization, data analysis and optimization techniques makes the proposed schema suitable for highly constrained problems through the construction of a robust process design tool that can achieve higher consistency by the optimization procedure and better conceivable results through the synchronous data analysis and visualization (Wang, Salhi and Fraga, 2004).

Wibig (2013) proposed a method for automated task scheduling in business processes models. In particular, the proposed method combines Petri nets to model the processes and simulate their execution, with dynamic programming for the reduction of the computational complexity by reconstructing only the parts of the model that have been changed. Finally, the simulation-based optimization is conducted by the Non-dominated Sorting Genetic Algorithm II to find the Pareto optimal solutions. Hofacker and Vetschera (2001) conducted a comparative analysis of three different mathematical techniques in regard to optimization of business process design problem: mixed integer mathematical programming, several variants of a direct branch and bound search strategy and genetic algorithms.

Additionally, they introduced (Hofacker and Vetschera, 2001) an appropriately adapted for each technique formal quantitative representation of business process, which is utilized by the corresponding analysis method for the determination of the optimal designs with respect to various min/max type objective functions. Concerning evolutionary algorithms, their computational experiments indicate a weak performance due to the difficulty of maintaining feasibility in a tightly constrained problem (Hofacker and Vetschera, 2001). Their formal model is hard to produce feasible solution because of the high number of constraints and mathematical formulations that implicates (Vergidis, Tiwari and Majeed, 2007). In the context of evolutionary computing, Vergidis, Tiwari and Majeed (2006a;2006b) proposed a framework that extends the approach of Hofacker and Vetschera, using the multi-objective evolutionary algorithms Non-Dominated Sorting Genetic Algorithm II (NSGA2) and Strength Pareto Evolutionary Algorithm II (SPEA2), in order to construct an automated method for the multi-objective optimization of business processes.

The proposed framework defines formally a generic business process model to guarantee that the pursued optimization is repeatable and verifiable. The optimization procedure is performed by NSGA2, which is applied on a predefined library of candidate activities and selects the optimum ones in the appropriate sequence by defining their starting times (Vergidis, Tiwari and Majeed, 2006b). The experimental results indicate that although the highly constrained nature of process optimization problem, the fragmentation of the searching space can lead to the production of several alternative processes that meet the optimization requirements. With a later work Vergidis, Tiwari and Majeed (2007) applied their framework to a larger number of objectives, focusing on the tasks that compose a business process rather than the process design itself.

Specifically, they introduced the concept of composite business process as a set of collaborative tasks in the appropriate sequence in order to achieve a business goal. From this point of view, each consisting task can be represented by a set of quantitative attributes such as cost, duration, quality, and flexibility, which can be used for the calculation of a quantitative evaluation. Consequently, task attributes can be mapped to the corresponding process attributes and provide an evaluation for the composite process via aggregation functions (Vergidis, Tiwari and Majeed, 2007). Based on this concept, Vergidis, Tiwari and Majeed formulated multi-objective optimization problems and then conducted a comparative application of the popular evolutionary multi-objective optimization algorithms NSGA2, SPEA2 and MORSO, in an attempt to generate optimum business process designs.

According to their approach, the process optimization can be defined as the selection and the proper sequencing of alternative tasks in a fixed process design that aims to satisfy specific predefined criteria. These criteria can be formulated as objectives or constraints in the base of process attributes. Considered as constraints, the criteria express the limit or the optimum value of the attributes, while considered as objectives, they must be satisfied as much as possible to maintain design's feasibility (Vergidis, Tiwari and Majeed, 2007). Vergidis and Tiwari (2008) proposed an optimization framework that applies evolutionary algorithms on quantitative models of business processes and generates diverse optimized process designs on the basis of predefined requirements. Specifically, they define business process design and attributes optimization as the problem of constructing feasible process design with optimum attribute values.

Design's feasibility is determined by the process resource requirements and the connectivity patterns of the participating tasks. The proposed framework introduces a quantitative representation of business processes and involves the application of genetic algorithms Evolutionary Multi-objective Optimization Algorithm (EMOOA) and NSGA2. The experimental application of the proposed framework on a variety of problems, demonstrates that despite the high complexity a satisfactory number of optimized alternative designs can be produced. An extension to the above framework proposed by Tiwari, Vergidis and Turner (2010). This extended framework involves a quantitative representation of business processes, an algorithm that composes feasible process designs and a set of optimization algorithms for the generation of diverse optimized designs constructed on the basis of predefined requirements (Tiwari, Vergidis and Turner, 2010). Their aim was to add the potentiality of selection the designs with the optimum objective values for business processes with large size. Another extension of the evolutionary optimization framework of Vergidis and Tiwari (2008) presented by Georgoulakos et al. (2017), who introduced an additional per-processing layer for the refinement of the predefined process requirements.

Specifically, the proposed extended framework utilizes a quantitative formulation of business processes, a composition algorithm that examines the feasibility of the constructed process designs, an additional step for the per-processing of the library of the available tasks and the evolutionary optimization algorithms NSGA2, SPEA2, PESA2 and PAES. The aim of this framework is to generate an arbitrary number of diverse optimized designs constructed on the basis of predefined requirements that have been refined by the per-processing layer (Georgoulakos et al., 2017). The pursued goal of per-processing is to reduce the computational complexity of the problem and to improve the performance of the evolutionary algorithms. In particular, the targets of the suggested per-processing step are (Georgoulakos et al., 2017):

2. The removal of tasks that require an input resource that belongs to the set of the required outputs of the entire business process.
3. The removal of tasks that produce an output resource that belongs to the set of the initial business process inputs.
4. The removal of tasks that require an input resource that is not produced by any other task or is not contained in the set of the initial business process inputs.
5. The removal of tasks with a single output resource that is not required as input by any other task.
6. The removal of tasks which are dominated by their alternative. Alternatives are considered the tasks that have identical input and output resources and distinguished only by their attribute values.

Using analogous rationale, Ahmadikatouli and Aboutalebi (2011) proposed a novel algorithmic methodology, which models a business process with Petri nets and introduces a formal quantitative structure for its optimization, considering a vector of evaluation criteria such as time, cost, quality of products and queue length. The proposed methodology considers the process representation as a dynamic system that is consisted by two components, a vector for the design and a matrix for the evaluation criteria. Subsequently, it utilizes a genetic algorithm to determine the optimized process design and a number of potential alternatives. The novelty is that instead of using the classical genetic operations crossover and mutation, the algorithm uses the defined by Ahmadikatouli and Aboutalebi (2011) operations parallelization and merge respectively. Parallelization operation is based on the sense that sometimes a complex activity might act as performance bottleneck, thus splitting it into smaller paralleled activities with the same input and output could probably accelerate the execution and improve the efficiency. On the contrary, merge operator tries to achieve execution acceleration and efficiency improvement by merging two or more very simple activities into one more complicated (Ahmadikatouli and Aboutalebi, 2011).

2.2.3.3 Data mining approaches

In the relevant literature, there is notable research effort that focuses mainly on data analytics and the comprehensive exploitation of the data generated during process execution. The aim is the development of prescriptive methods to transform analysis results into improvement actions. Groger, Schwarz and Mitschang (2014) suggested a data-mining technique for the recommendation-based business process optimization (rBPO) as the base of an adaptive and continuous optimization procedure. rBPO exploits descriptive analytics from the data that is generated during business process execution and recommends remedial actions for the avoidance of the predicted deviations from the predefined metric standards.

In the core of the system there is a holistic process warehouse, which initially integrates the transactional data that is collected from a variety of sources during process execution and then employs classification techniques pursuing real-time prediction and recommendation generation (Groger, Schwarz and Mitschang, 2014). Niedermann and Schwarz (2011) developed the deep Business Optimization Platform (dBOP), which integrates a layer of formalized optimization patterns and techniques, with a layer of heterogeneous data analysis and integration capabilities for specific situations. The dBOP is consisted by three functionally sequential layers. The data layer manipulates the integration of heterogeneous data that is collected from various sources during process execution. The analytics layer automatically analyses the integrated data on the basis of the graph analysis of the employed optimization patterns, using standardized metrics and data mining techniques.

Finally, the optimization layer combines an optimization engine and the optimization patterns that are stored in the pattern catalog to discover and apply process improvements (Niedermann and Schwarz, 2011).The pattern catalog is a collection of formal patterns classified according to a set of quantitative and qualitative criteria. Namely, the type of changes they implement, the level of their contribution towards the fulfillment of the optimization goal, the process stage during which they can be applied, the set of constraints that they must satisfy and finally, the required data in order to be involved in the process design (Niedermann and Schwarz, 2011).Gerke, Petruch and Tamm (2010) presented a solution for the optimization of service delivery through the continual process improvement. Their approach involves a data mining methodology for the automated inference of process knowledge from a set of individual process instances. In particular, the execution of these instances is continuously monitored and recorded in event logs appropriately formed in in order to be processed by the data mining algorithms provided by the process mining framework ProM2.

Process mining can discover information about the performance, the weaknesses and the improvement potential of a process (Gerke, Petruch and Tamm, 2010). On the basis of the knowledge derived from process mining, an internal benchmarking is conducted against predetermined key performance indicators. The benchmarking can reveal the steps of the process that prevent the fulfillment of the business objectives (Gerke, Petruch and Tamm, 2010). Finally, a considerable number of algorithms exist for the optimal scheduling of sub-processes of a more complex process. Sub-processes can be combined in many different temporal and spatial sequences and the problem is to find the sequence that optimize a set of predefined criteria, considering relation constraints between sub-processes. However, the fact that there is an infinite number of components that can be used to construct an administrative process, makes exponential the computational complexity of searching in process elements domain (Hofacker and Vetschera, 2001).

2.3 Research Gap

From the description of the methodologies and frameworks that have been developed to solve the problem of business process optimization can be discerned a common characteristic. In principle, all of them try to formulate the rules for the implementation of a comprehensive and effective search within the domain of business tasks, in order to discover the consisting parts that can construct the optimum business process on the basis of predefined requirements. The majority of the methods are based on graphical paradigms, evolutionary methodologies or data mining techniques while is observed a complete absence of methods that are founded entirely or partially on constraint logic programming (CLP). However, CLP is by nature a searching technique in the domain of decision variables under the limitation of predefined constraints. Thus, CLP can offer the computational environment for the confrontation of complex combinatorial problems such as business process optimization, which can be treated as a COP problem with precedence and resources constraints that must be optimized in respect to a set of evaluation criteria.

Therefore, the aim of the present thesis is to investigate the applicability of CLP to the field of business process optimization by the design and development of a framework that can compose optimized complex business processes from simpler business tasks in the base of predefined quantitative and qualitative evaluation criteria. For the achievement of this goal, the challenge is to find the appropriate quantitative representation of entities such as tasks, resources and business processes, which simulates adequately the real world and is suitable to be used in a CLP program. Additionally, must be defined the qualitative and quantitative evaluation criteria, along with the objective function that incorporates them as metric for the optimization procedure. Finally, technical challenges arise by the effort to integrate heterogeneous and incompatible technologies such as relational databases, object-oriented programming, visualization techniques and logic programming within the functional boundaries of a single software system.

2.4 Summary

The second chapter presents the semantics of the terms used in thesis and explains their definitions that derived from the relevant literature and adopted as the most comprehensive. Subsequently, it briefly outlines the major methodologies that have been developed in respect to the problem of processes optimization within the domain of business administration, presents the major principles of the CLP paradigm and explains in detail the formulations of the ERCPSP as a COP problem. Finally, it concludes the complete absence of methods that are founded entirely or partially on CLP, justifying the investigation of the applicability of CLP to the field of process optimization, through the design of a framework that composes optimized processes from simpler tasks in the base of predefined evaluation criteria. The next chapter presents the methodological approach utilized for the fulfillment of the distinct challenges and the implementation of the aforementioned complex software system.

3 Mathematical foundation

The third chapter is devoted to the comprehensive presentation of the principles and techniques that constitute the mathematical foundation of the developed framework and can guarantee the structural consistency and rational robustness of the proposed optimization procedure. More specific, the first sub-chapter describes the classic formulation of the Resource Constrained Scheduling Problem, which is the mathematical basis of the suggested business process optimization algorithm. The second sub-chapter analyses in details the principles of the Constraint Logic Programming, which is the main mathematical paradigm that is used for the implementation of the optimization procedure. Finally, the third sub-chapter explains the function of the binary search technique, which is of crucial importance for the reduction of the computational complexity of the optimization algorithm.

3.1 Resource Constraint Scheduling Problem (RCSP)

In general, the well-known Resource Constrained Project Scheduling Problem assumes that a process is consisted by a set of tasks of known duration and resource requests from a predefined set of available resources of limited availability. The tasks are linked by precedence relations and the objective is the construction of a feasible schedule with the minimum duration in respect to the precedence relations and the resources availability (Artigues, 2008;Kolisch, 1996). More formally, the RCPSP problem is a combinatorial optimization problem that can be defined by a discrete solution space X and a subset of feasible solutions $Y \subseteq X$ in the base of an objective function $F: Y \rightarrow \mathbb{R}$. The aims is the discovery of a feasible solution $y \in Y$ such that $F(y)$ is optimized (Artigues, 2008). In mathematical formalism, RCPSP is defined by a tuple *(T, DR, P, R, A, DM)* where (Artigues, 2008):

1. $T=\{T_0,\ldots,T_{n+1}\}$ is the set of tasks that constitute the schedule. The dummy tasks T_0 and T_{n+1} represent by convention the start and the end of the schedule respectively, thus the set of non-dummy tasks is the $T=\{T_1,\ldots,T_n\}$.
2. The vector DR in $\mathbb{N}^{n+2}$ represents the durations of the tasks, thus dr_i is the duration of activity T_i with special values $dr_0 = dr_{n+1} = 0$.

3. The precedence relations are contained in the set P. The pair $(T_i, T_j) \in P$ denotes that the task T_i precedes the task T_j. The assumption is that T_0 is a predecessor and T_{n+1} is a successor of all the other tasks.
4. The available resources are contained in the set $R=\{R_1,\ldots,R_q\}$.
5. The availability of the resources are represented by the vector A in $\mathbb{N}^q$ such that A_k denotes the availability of the resource R_k.
6. The demands of tasks for resources are contained in the $(n+2)\times q$ integer matrix DM, such that dm_{ik} denotes the amount of resource R_k used per time period during the execution of A_i.
7. A schedule is a point S in $\mathbb{R}^{n+2}$ such that S_i represents the start time and E_i the end time of the task A_i , with $E_i = S_i + dr_i$. It is assumed that $S_0 = 0$ as a reference point for the beginning of the schedule.
8. A solution S is feasible if it is compatible with the precedence constraints (1) and the resource constraints (2), where $T_t=\{T_i \in T \vee S_i \leq t < S_i + dr_i\}$ represents the set of non-dummy activities in process at time t.

$$S_j - S_i \geq dr_i, \forall \left(T_i, T_j\right) \in P (1)$$

$$\sum_{T_i \in T_t}^{\square} dm_{ik} \leq A_k, \forall\ R_k \in R, \forall\ t \geq 0 (2)$$

Using the above notation the RCPSP can be defined as the "*problem of finding a non-preemptive schedule S of minimal makespan S_{n+1} subject to precedence constraints (1) and resource constraints (2)*" (Artigues, 2008). The RCPSP is considered as a generalization of the Static Job Shop Problem and Blazewicz, Lenstra and Kan (1983) proved that belongs to the class of NP-hard problems. In order to extend RCPSP, we consider that the set $T=\{T_1,\ldots,T_n\}$ is unknown and we have to search if it exists as a subset of a tasks library $P_{library}$. The key point of the search is to discover the initially unknown precedence relations between the tasks of T, in the base of their input/output resources combination and the predefined mutual exclusions that might exist. A feasible solution is a schedule of interconnected tasks $\in P_{library}$, which consumes a predefined set of input resources, produces a predefined set of output resources and satisfies the constraints about the resources availability.

3.2 Constraint Logic Programming (CLP)

Logic programming (LP) is a programming paradigm based on mathematical techniques for automated proof of theorems and is foundations lay on the resolution principle and the notion of unification. The resolution proof procedure, which was introduced by Alan Robinson in 1965, can construct proofs for axioms in the form of first-order logic formulas and answer positively or negatively to a question (Apt and Wallace, 2007). Later in 1974, Robert Kowalski proposed a modification that enables the resolution method to overcome its limitations and to compute answers to a question (Apt and Wallace, 2007). The new version allows the construction of a substitution that satisfies the original formula and this substitution is considered as the result of a computation (answer set substitution). The revised resolution method led to the introduction of a new programming paradigm, known as logic programming (Apt and Wallace, 2007). Constraint programming (CP) is another mathematical method for solving combinatorial search problems. CP consists of the declarative formulation of constraints on the feasible solutions for a set of decision variables and the selection of a search strategy (Rossi, van Beek and Walsh, 2006).

The fundamental concept of CP is the notion of constraint and it has been used extensively in computer science and artificial intelligence the last decades (Rossi, van Beek and Walsh, 2006). During the same period, in the context of the aforementioned scientific fields was identified the problem of constraint satisfaction and formulated the concept of constraint propagation for the confrontation of the combinatorial explosion when solving constraint satisfaction problems using a top-down search. The term top-down search refers to the technique of constructing a final solution by the systematic extension of a partial solution through the application of constraints (Niederlinsky, 2014). In particular, this type of search combines a branching strategy along with constraint propagation in order to split a complex logic problem into an arbitrary number of simpler, the union of which is equivalent to the initial complex problem. This procedure creates during execution the so-called search tree (Figure 2.2) where the top-down search technique is applied, considering each path as a vector of decision variables (Apt and Wallace, 2007).

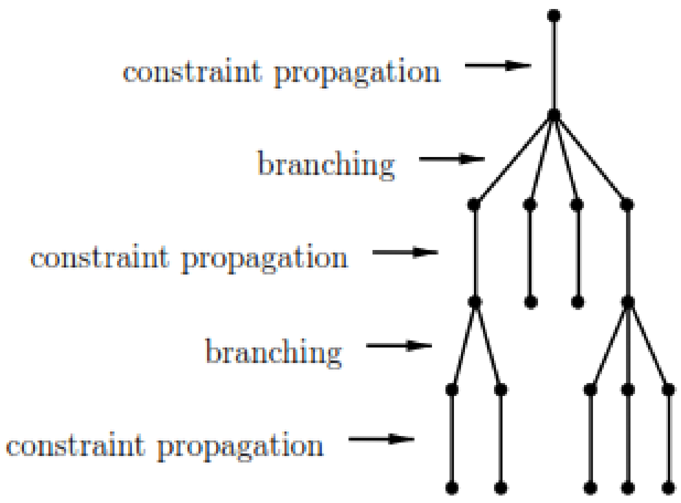


Figure 3.1.Search tree (Apt and Wallace, 2007, p.112)

The most commonly used top-down search techniques are backtracking search, a variation of depth-first search and branch and bound search for optimization problems (Niederlinsky, 2014). Backtracking search starts at the root, moves towards the first leftmost descendant node-variable and assigns a value from its domain. If there are more than one values to be assigned, it creates a choice-point saving the state of the search. The procedure continues until the visited node is a leaf or a violation of a constraint arises. Then search goes back to the closest ancestor where the last choice-point was created and moves toward the next more left descendant. When all of its descendants have been visited, the choice-point is removed and the search returns to the choice-point of the higher level according to the described rationale. Backtracking search finishes when the control is back to the root and the possible value assignment combinations of the decision variables have been tested (Apt and Wallace, 2007). Of course, problem independent heuristic search procedures, have been proposed that increase the efficiency of the CP approach significantly on a number of problems, with the most celebrated being the first-fail heuristic.

Branch and bound is a general paradigm of combinatorial optimization, designed to find global optima of non-linear objective function that are limited by non-linear constraints (Apt and Wallace, 2007). However, in practice is used mainly for linear objective function under linear constraints. The main difference from backtracking search is that branch and bound utilizes an objective function and compares its best so far value with its current value (Apt and Wallace, 2007). Specifically, when the search has run through the leftmost branch of the tree saves the value of the objective function as the best so far bound and the corresponding variables' vector as the optimum solution. The search control moves to the next branch and if upon reaching its leaf, the current value of the objective function is better than the best so far, is saved as the new best so far and the corresponding variables' vector as the optimum solution. Otherwise, if the current value is worse than the best so far or while running through the branch reaches an intermediate node and the current value of the objective function is already worse than the best so far, then the search stops and the control moves to the next branch.

The remarkable concept in this case is that CP program can be developed as an LP program and executed in a LP platform that is properly extended by the concept of constraints. Indeed, CP constraints and LP predicates are both mathematical relations, in both CP and LP decision variables considered unknown and backtracking is the main top-down search technique. The above combination resulted to the constraint logic programming (CLP) (Apt and Wallace, 2007). In mathematical formalism, CLP is defined as a method for solving constraints satisfaction problems (CSP) with the following characteristics (Niederlinsky, 2014):

1. They involve a finite set of integer variables $S=\{X_1, X_2, \ldots, X_n\}$ with values from the corresponding finite domains $\{D_1, D_2, \ldots, D_n\}$.
2. There is a set of constraints between variables. The *ith* constraint $C_i(X_{i1}, X_{i2}, \ldots, X_{ik})$ between *k* variables from S can be expressed as relation (equation, inequality or a subroutine), which is defined as a subset of the Cartesian product $D_{i1} \times D_{i2} \times \ldots \times D_{ik}$ that determines the values of the variables.
3. A CSP solution is any assignment of domain values to variables that satisfies all the constraints.
4. When a CSP solution additionally optimize an objective function, the problem is referred as constraint optimization problem (COP) and its solution as optimum solution.

Considering the definition of the RCPSP problem as presented in 3.2, it can be treated as a COP problem with precedence and resources constraints that must be optimized in respect to a set of evaluation criteria. This ascertainment encourages the research about the applicability of CLP for the confrontation of the RCPSP problem and maybe some of its variations.

3.3 Binary search technique

The binary search is a recursive algorithm designed to apply on a sorted group of elements in order to ascertain if it contains a target element *T* or not (Knuth, 1998). The idea is to compare *T* with the middle element of the sorted group. In case there are not equal, the procedure is repeated with the half of the group where T is possible to exist, ignoring the other half. This iteration continues until just one element A_m of the group is left. If $T = A_m$ then the target element is found, otherwise the target element is not contained in the group (Wikipedia contributors, 2019). In a more formal mathematical notation, given a group *A* of *n* sorted elements $A_1 \leq A_2 \leq \cdots \leq A_{n-1}$ and a target element T the steps of a general form of the binary search algorithm are the following (Wikipedia contributors, 2019):

1. Set left bound *L* to 1 and right bound *R* to *n*
2. If $L = R$ go to step 6
3. Set A_m (the middle element) to the ceiling of $(L + R) / 2$, which is the least integer greater than or equal to $(L + R) / 2$
4. If $A_m < T$, set *R* to $m - 1$ and go to step 2
5. Set *L* to A_m and go to step 2
6. Now $L = R$ and if $L = R = T$, return true, otherwise return false

Considering that every iteration of the binary search splits the sorted group of elements into half, the binary logarithm has a logarithmic computational complexity $O(\log_2 n)$, in the worst case it requires no more than $\lfloor \log_2 n \rfloor + 1$ comparisons and in average it makes about $\log_2 n - 1$ comparisons for a successful search, where *n* is the size of the group (Knuth, 1998). In the optimization algorithm that will present in this thesis, we divide the available tasks into twenty subgroups ordered according to the id of their first input resource. In order to locate a specific subgroup using binary search is utilized a vector of nineteen values, which represent the id of the first input resource of the last task of the groups 2 to 19. Figure 3.2 shows the complete searching tree that is created for $n = 20$, where we can observe that the longest path is composed by $\lfloor \log_2 20 + 1 \rfloor = 5$ comparisons.

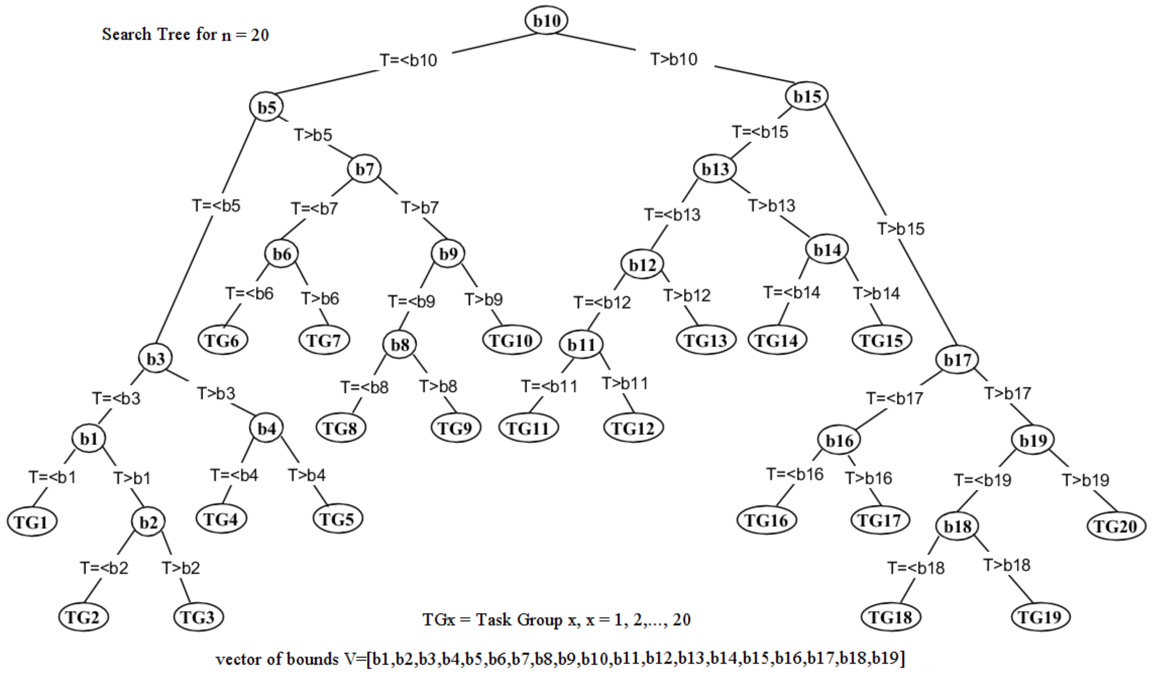


Figure 3.2.Binary search tree for n = 20

3.4 Summary

In the third chapter was presented the mathematical background where this thesis is based on, namely the Resource Constraint Scheduling Problem formulation, the principles of the Constraint Logic Programming paradigm and the functionality of the binary search technique. The reader, equipped with the aforementioned information will be able to follow easily the reasoning of the design and development of the proposed process optimization procedure.

4 Methodology

In the context of the present thesis, a business process can be defined as a collection of activities arranged in a sequence in order to perform a specific operation (Vergidis and Tiwary, 2008). In more detail, the business process' elements are the participating tasks, the task attributes, their input and output resources and the connectivity relations in terms of precedence or concurrent execution constraints (Tiwari, Vergidis and Turner, 2010). The resources are physical or information objects flowing through the system and can be distinguished in initial input resources, required output resources and the intermediate produced resources (Hofacker and Vetschera, 2001). The tasks are transformation procedures that consume resources as inputs and produce other resources as outputs (Hofacker and Vetschera, 2001). Finally, the potential connectivity patterns are constrained by the requirement that resources must be contained in the set of initially input resources or the set of intermediate produced resources before they can be consumed by other tasks (Hofacker and Vetschera, 2001).

Therefore, the challenge of the entire process optimization is manipulated through the quantitative representation of tasks by a set of five criteria, namely cost, revenue, duration, quality and flexibility (Hofacker and Vetschera, 2001;Vergidis and Tiwary, 2008;Dumas et al., 2013). Considering that the individual attributes of the participating tasks can be mapped to the entire process attributes, the evaluation of a business process can be achieved by aggregating the evaluations of the consisting task attributes using appropriate functions (Vergidis, Tiwari and Majeed, 2007). Thus, the objective of the optimization procedure is to search for the tasks that can construct the feasible process design with the best evaluation in the base of the aforementioned evaluation criteria.

4.1 Conceptual analysis of the evaluation criteria

The execution time of a process can be separated in processing time, which is the time for the elaboration of input resources and in waiting time, which is the time that a process spends in idle mode (Dumas et al., 2013). The cost is considered as a function of the sum of the values of consumed resources and similarly, the revenue is the sum of the value of the resources that are produced by a process (Dumas et al., 2013). The notion of process quality can be distinguished between external and internal quality. The external quality can be measured as the client's satisfaction from the outcome of the process, namely, the extent to which the client's expectations are met by the delivered product.

On the contrary, the internal quality of a process refers to the level that the participants feel that control the performed tasks and the working variation they experience. Additionally, a measure of the internal quality could be whether the participation within the context of the process is challenging in terms of self-improvement and target achievement (Dumas et al., 2013). Finally, the flexibility of a business process is closely related to the issue of change and can be defined as the capability to react to unexpected processing diversions. That is, the ability to handle variations while executing a specific business process setting, the easiness to alter the structure and responsiveness of the process according to market requirements and the ability to reallocate responsibilities, work flows and employee's roles (Dumas et al., 2013).

4.2 Problem formulation

The foundation of the optimization procedure upon a formal mathematical model ensures the structural consistency and can guarantee the construction of feasible process designs by a repeatable and verifiable approach, which aspires to optimize a set of objectives functions through the satisfaction of a set of predefined relational constraints (Vergidis, Tiwari, and Majeed, 2006b). In the context of the present thesis, it is considered that a complex business process can be constructed by a set of *t* simple tasks that are stored in a task library $P_{library}=\{p_1, p_2, \ldots, p_t\}$. Each task *i* consumes resources of specific type and quantity as input I_i and produces resources of specific type and quantity as output O_i. All of the *m* available resources are stored in a resource library $R_{library}=\{r_1, r_2, \ldots, r_m\}$, and the following $I_i \subseteq R_{library}$ and $O_i \subseteq R_{library} \forall i \in P_{library}$ hold. Additionally, each task *i* is characterized by a vector of five measurable attributes $a_i=\{a_{i1}, a_{i2}, a_{i3}, a_{i4}, a_{i5}\}$, namely cost, duration, revenue, quality and flexibility respectively, which can be used for its quantitative evaluation. The tasks' attributes that construct a complex business process are mapped to its corresponding attributes via appropriate aggregate functions and min/max relations depending on the attribute's nature. This quantitative evaluation is a key factor of the optimization procedure.

The business process optimization is considered as a NP-hard problem due to the non-linear, non-convex and often discontinuous nature of the involved mathematical models (Wang, Salhi and Fraga, 2004), meaning that the computational efficiency depends on the size of the examined instance of the problem (Georgoulakos et al., 2017). For the reduction of the high complexity and the improvement of performance, the task library $P_{library}$ is formed using an index schema that groups the available tasks in non-overlapping categories. Following this strategy, the search domain of a specific instance of the optimization problem is only a small subset of $P_{library}$, that is a certain category only, leading to a significant reduction of the variables' combinations that must be produced and evaluated by the CLP program and thus, to improvement of its computational efficiency. The parameters that are involved in the problem formulation are described in Table 4.1.

Table 4.1.Mathematical model parameters

Parameter	**Description**
$P_{library}$	Library of available tasks
C_x	Category of tasks according to index schema, $C_x \subseteq P_{library}$
$R_{library}$	Library of available resources
n	Number of tasks in category $C_x \subseteq P_{library}$
m	Number of available resources in $R_{library}$
r_j	Resource of type *j* in $R_{library}$
I_{BP}	Set of initially available input resources
O_{BP}	Set of initially requested output resources
R_{IP}	Set of the intermediately produced resources
I_i	Set of input resources of task *i*
O_i	Set of output resources of task *i*
$in_i[r_j, q_j]$	Quantity q_j of resource *j* as input to task *i*
$out_i[r_i, q_i]$	Quantity q_j of resource *j* as output of task *i*
$ip[r_j, q_j]$	Quantity q_j of intermediately produced resource *j*
ct_i	Cost of task *i*
dr_i	Duration of task *i*
rv_i	Revenue of task *i*
ql_i	Execution quality of task *i*
fx_i	Flexibility of task *i*
$pr_i[p_1, p_2,..., p_v]$	Set of *v* tasks that directly preceding task *i*
$mex_i[p_1, p_2,..., p_u]$	Set of *u* tasks that mutually excluded with task *i*
$p_i(id_i, name_i, ct_i, rv_i, dr_i, ql_i, fx_i, in_i[r_j, q_j], out_i[r_i, q_i], mex_i[p_1, p_2,..., p_v])$	Representation of task *i*
BPM(Tasks, cost, revenue, duration, quality, flexibility, evaluation, Starts, Finns, Connectivity)	Representation of the constructed optimized business process

The resources that participate to the construction of a specific business process design can be distinguished in the set I_{BP} of the initially available input, the set O_{BP} of the requested output by the complex constructed process and the set R_{IP} of the resources, which are produced by the tasks that are included in the complex design. The initial aim of the optimization procedure is to define all the feasible interconnection patterns between the set of tasks that belong to a certain category $C_x \subseteq P_{library}$, for the construction of a complex business process that consumes all the resources contained in $I_{BP} \cup R_{IP}$ and produces all the resources contained in O_{BP}. However, the primary target is to distinguish among the feasible designs those with the best values for their attributes $a = \{ct, dr, rv, ql, fx\}$. Therefore, the mathematical model defines five objective functions, namely:

$$F(ct) = \sum_{i=1}^{k} ct_i \rightarrow min (1)$$

$$F(dr) = \sum_{i=1}^{k} max[par_i] \rightarrow min (2)$$

$$F(rv) = \sum_{i=1}^{k} rv_i \rightarrow max (3)$$

$$F(ql) = min[ql_1, ql_2, \ldots, ql_n] \rightarrow max (4)$$

$$F(fx) = min[fx_1, fx_2, \ldots, fx_n] \rightarrow max (5)$$

If we transform the objective functions (1) and (2) as follows:

$$F(ct) = \sum_{i=1}^{k} ct_i \rightarrow min \Leftrightarrow -\sum_{i=1}^{k} ct_i \rightarrow max (1)$$

$$F(dr) = \sum_{i=1}^{k} max[par_i] \rightarrow min \Leftrightarrow -\sum_{i=1}^{k} max[par_i] \rightarrow max (2)$$

the five objective function can be linearly combined into a single objective function to be optimized:

$$F(opt) = -F(ct) + F(rv) - F(dr) + F(ql) + F(fx) \rightarrow max$$

Considering that k is the number of participating tasks in the construction of the optimized process, then the first objective function minimizes the total cost and the second minimizes the total duration of the optimized process. The total duration is the addition of the maximum duration of the groups of tasks that are executed in parallel.

The third objective function maximizes the total revenue obtained by the participating tasks. For the qualitative attributes execution quality and flexibility, the mapping to the corresponding attributes of the constructed process is based on the concept of the common sense that the qualitative characteristics of complex structures are specified exclusively by the consisting parts with the minimum value.

Therefore, the fourth and fifth objective functions pursue the maximization of the minimum value of execution quality and flexibility respectively. The parameter $pr_i[p_1,p_2,...,p_v]$ is used to denote the ancestor-descendant relations between tasks that participate to the design and concerns the groups of tasks that can be executed in parallel. Thus, the optimized business process design can be considered as a set of temporally sequential groups, which contain parallel-executed tasks. The constraints of the logic model limit the optimization procedure and guide the construction of consistent and feasible business processes with optimized values for their attributes. The complete set of the logic formulas that formulate the constraints of the logic model is the following:

1. $\forall p_i \in BPM([r_j,q_j] \in I_i \wedge [r_j,q_a] \in I_{BP} \cup R_{IP} \wedge q_j \leqslant q_a)$

   A task participates in process design when all its input resources are available in equal or bigger quantities.
2. $\forall [r_j,q_j] \in R_{IP} \cup O_{BP} (\exists p_i \in BPM([r_j,q_a] \in O_i \wedge q_a \leqslant q_j))$

   All the resources that belong to the set R_{IP} of the intermediately produced resources or to the set O_{BP} of the output of a feasible process design must be produces as output by at least one participating tasks in equal or less quantity.
3. $\forall [r_j,q_j] \in R_{IP}$

   $(\exists [p_i,...,p_k] \in BPM([r_j,q_i],...,[r_j,q_k] \in O_i \cup ... \cup O_k \wedge q_j = q_i + ... + q_k))$

   The quantity of each intermediately produced resource must not exceed the sum of quantities, which are produced by the participating tasks that have as output the specific resource.
4. $\forall p_i, p_j \in BPM(\neg(i=j))$

   A task can be used once to avoid cyclic dependencies between tasks. This can occur when one task produces the input resources of another task, which in turn produces the input resources of the former task.
5. $\forall [r_j,q_j] \in O_{BP}$

$(\exists([p_i,\dots,p_k]\in BPM \wedge [r_j,q_i],\dots,[r_j,q_k]\in O_i\cup \dots \cup O_k \wedge q_j=q_i+\dots+q_k))$

The quantity of each final output resource must not exceed the sum of quantities, which are produced by the participating tasks that have as output the specific resource.

6. $\forall p_i,p_j\in BPM(p_i \rightarrow p_j \Rightarrow \exists[r_k,q_i]\in O_i \wedge \exists[r_k,q_j]\in I_j)$

   Two participating tasks p_i, p_j are connected with the preceding- succeeding relation $p_i \rightarrow p_j$ when the task p_i produces as output an input resource of the task p_j.

7. $\forall p_i,p_j\in BPM(p_i \otimes p_j \Rightarrow (e_i \leqslant s_j)\vee(e_j \leqslant s_i))$

   Two participating tasks p_i, p_j are connected with the mutual exclusion relation $p_i \otimes p_j$, when the end time e_i of the task p_i is earlier or equal than the start time s_j of the tasks p_j or the end time e_j of the task p_j is earlier or equal than the start time s_i of the tasks p_i.

8. $\forall p_i\in BPM(\varnothing \rightarrow p_i \Rightarrow \forall[r_j,q_j]\in I_i(\exists([r_j,q_a]\in I_{BP} \wedge q_j \leqslant q_a)))$

   A participating task has not a preceding task(s), when all of its input resources are contained into the set I_{BP} of initially available resources in equal or bigger quantities.

9. $\forall p_i\in BPM(p_i \rightarrow \varnothing \Rightarrow \forall[r_j,q_j]\in O_i(\exists([r_j,q_a]\in O_{BP} \wedge q_j \leqslant q_a)))$

   A participating task has not a succeeding task(s), when all of its output resources are contained into the set O_{BP} of the required resources in equal or bigger quantities.

10. $\forall p_i,p_j\in BPM(p_i \| p_j \Rightarrow (s_i \leqslant s_j \wedge e_i > s_j)\vee(s_j \leqslant s_i \wedge e_j > s_i))$

    Two participating tasks p_i, p_j are connected with the parallel execution relation $p_i \| p_j$, when the start time s_i of the task p_i is earlier or equal than the start time s_j of the tasks p_j and the end time of the task p_i is later than the start time of the task p_j or vice versa.

The algorithm terminates when all of the following termination conditions are fulfilled:

1. All the resources that are contained in O_{BP} have been produced in the required quantities.
2. All the initially input resources that are contained in I_{BP} have been consumed.
3. All the intermediately produced resources that are contained in R_{IP} have been consumed.

4.3 Task library

The proposed software system includes a relational database, which contains an arbitrary number of tasks and resources. This database acts as the library of the fundamental building blocks that will be used for the construction of the entire optimized business process. However, the most significant contribution of the database to the optimization procedure is the implementation of the first step of the data reduction, namely the selection of the candidate tasks that fulfill the requirements to participate to the construction of the optimized business process in respect to user's choices. Specifically, for the storage of the available tasks is utilized a hierarchical schema that is based on the Value Chain Analysis management concept which was developed by Michael Porter (1985). Value Chain (Table 4.2) is a conceptual collection of general classes, where theoretically can be classified all the processes that a company can perform in order to create value for its customers, to pursue competitive advantages and to attain higher profitability. Value Chain Analysis focuses on the systems and activities considering customers as the central principle rather than the functional departments or the organizational hierarchy (Porter, 1985). The utilization of this hierarchical schema for the organization of the task library, results a significant limitation of the size of the searching domain, so that the application of searching strategies can become computationally more effective.

Table 4.2.Porter's Value Chain

Inbound Logistics	Production	Outbound Logistics	Marketing & Sales	Service
Procurement				
Human Resources Management (HRM)				
Research and Technological Development (R&TD)				
Infrastructure				

According to Porter (1985), the processes that an organization can engage to transform the available resources to products, in general can be classified in five primary categories and four supporting categories that serve as auxiliary processes for the proper implementation of each primary process. In particular, the primary categories are:

1. **Inbound Logistics**: includes the processes that are required to receive, store and distribute inputs.

2. **Production**: includes the processes that are required to transform resources into products.
3. **Outbound Logistics**: includes the processes that are required to collect, store and distribute the output.
4. **Marketing and Sales**: includes the processes that inform customers about products, inducing them to purchase.
5. **Service**: includes the processes that required to maintain the effective functionality of the products after purchase.

Moreover, the supporting categories are:

6. **Procurement**: includes the processes for the acquisition of resources.
7. **Human Resources Management**: includes the processes that are involved in recruiting, hiring, training, developing, compensating and dismissing employees.
8. **Research and Technological Development**: includes the processes for the acquisition of the equipment such as hardware, software, machinery, tools, procedures and technical knowledge.
9. **Infrastructure**: includes the processes that interconnect the various departments of the company such as accounting, legal, finance, planning, public affairs, government relations, quality assurance and general management.

The structure of the relational model of the database and the developed programming routines (PL SQL triggers and functions) facilitate the preservation of the overall data consistency, utilizing the embedded consistency mechanisms of the DBMS to guard against possible violations of constraints or relationships. Furthermore, they offer a single update point by disseminating automatically the applied changes throughout the entire database and make easy to insert new data, alter the existing or delete the obsolete without any alterations to the rationale of the framework. The complete logical model of the hierarchy schema of the task library is a tree graph (Figure 4.1) where the tasks are located into the leafs according to the functional sector of the company that they belong. The category *General* introduced to store the tasks that can be used to more than one functional sectors and eliminates the need for task repetition. Responsibility of the user is to insert as parameters the functional sector that the process to be optimized belongs, the set of initially input resources and the set of desired output resources. Thus, the framework can limit the extent of the searching domain utilizing the aforementioned data reduction constraints, only to the tasks that are contained into the corresponding leaf, reducing significantly the complexity of the optimization procedure..

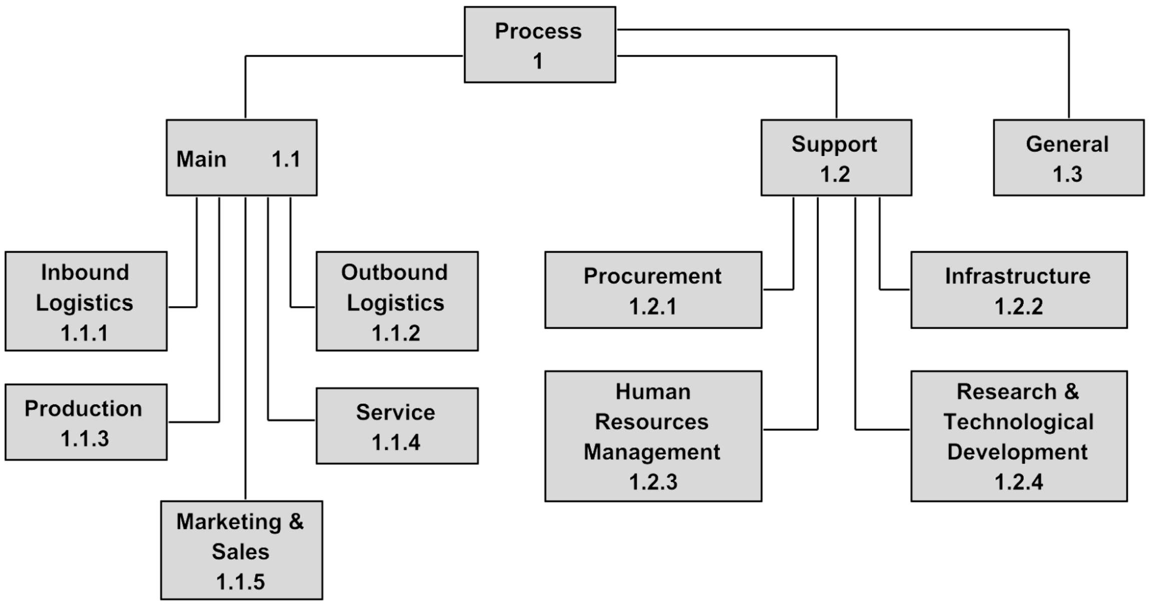


Figure 4.1.Task library hierarchy schema

The database was developed with the DBMS MySQL and is consisted by four relational tables as shown in the Entity Relation Diagram (Figure 4.2). The table *Category* indicates the functional sector where a task belongs according to the indexing schema that was described to the previous paragraph. The table *In_out_resource* relates each task with the resources that requires as input and produces as output, using the field *type* as indicator to denote if a resource is input or output and the field *quantity* to store the amount of the specific resource. Furthermore, three embedded triggers (Figure 4.2) disseminate and incorporate any changes (insert, update, delete) to the table *Process* according to the relationship constraint between the two tables. The table *Process* stores the partial quantitative representations of the available tasks. The value of the fields *duration*, *quality* and *flexibility* is predefined at storage time, while the value of the fields *cost* and *revenue* is calculated by the stored function *CalculateCostRevenue* (Figure 4.2) in order to maintain their updatability after a possible change of resources' cost. The table *Resource* contains the available resources that will be used for the construction of the optimized business process as input or output and the embedded trigger (Figure 4.2) disseminates the inserted value changes of the field *cost*, in order to preserve the accuracy of the fields *cost* and *revenue* of the table *Process*.

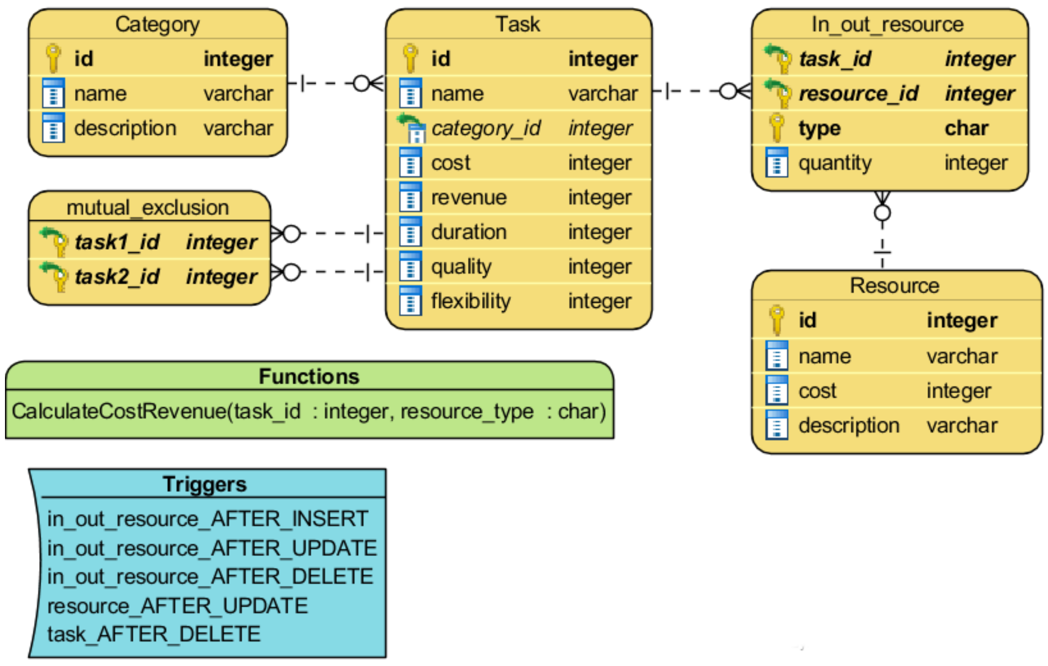


Figure 4.2.ER diagram of the database

4.4 Data reduction

One of the major problems for the effectiveness of the searching techniques is the high computational complexity when they have to elaborate very large domains. The manipulation of huge amounts of data can make the searching procedure impractical or even infeasible. For the confrontation of high data volume problem, the proposed system applies three preprocessing rules for data reduction. More specifically:

1. *The search domain is consisted only by tasks that belong to the functional sector defined by the user*. For the implementation of this rule, the proposed system borrows a data reduction technique that comes from the field of data mining, namely the Concept Hierarchy Generation (CHG) appropriately customized for the case. CHG specifies and groups portions of data in the base of conceptual relations and creates a hierarchical structure using the interrelationships between them (Han, Kamber and Pei, 2012). This data reduction rule is implemented by the logical model of the task library and for a specific instance limits the search domain only to a single task category.
2. *Rejection of tasks that require an input resource that is included in O_{BP}*. The production of the resources that belong to the requested optimized process output is one of the termination conditions of the optimization procedure. Thus, if a task

requires as input resources that are contained in O_{BP}, certainly will not be part of the optimized process design and can be excluded from the search domain (Georgoulakos et al., 2017). This rule is enforced by the Java module of the system by utilizing the appropriate SQL queries according to user's input.

3. *Rejection of tasks that produce an output resource that is included in I_{BP}.* The resources that belong to the initial input are considered as already available at the beginning of the optimization procedure and there is no need to be produced. Therefore, if a task produces as output resources that are contained in I_{BP}, certainly will not be part of the optimized process design and can be excluded from the search domain (Georgoulakos et al., 2017). This rule is also applied by the Java module by using the appropriate SQL queries according to user's input.

4.5 JAVA GUI and CLP program constructor

The initial interface of the framework is the control center of the application, where the user is provided with the necessary forms in order to insert the required data and adjust the appropriate parameters for the construction and implementation of the business process optimization procedure. Additionally, the interface contains in the background a set of routines that utilize the user input to acquire data from the database and to formulate the CLP program. Subsequently, it initiates the ECL^{i}PS^{e} platform by transmitting the constructed CLP program and appropriately transforms the returned calculation to be visualized by the visualization module. In more detail, the user has first to connect to the database using the credentials of a legitimate user's account (Figure 4.3, forms 1 and 2). The connection success or failure is denoted by a relevant pop up message. The procedural part is completed with the indication of the folder where the ECL^{i}PS^{e} platform is installed (Figure 4.3, form 3). On successful connection, the drop down list *Choose Process Category* (Figure 4.3, form 4) is automatically filled with the available task categories that were retrieved from the database. When the user chooses the desired category, the corresponding routine retrieves from the database the relevant to the tasks data using the appropriate SQL queries and stores it to a local temporary structure.

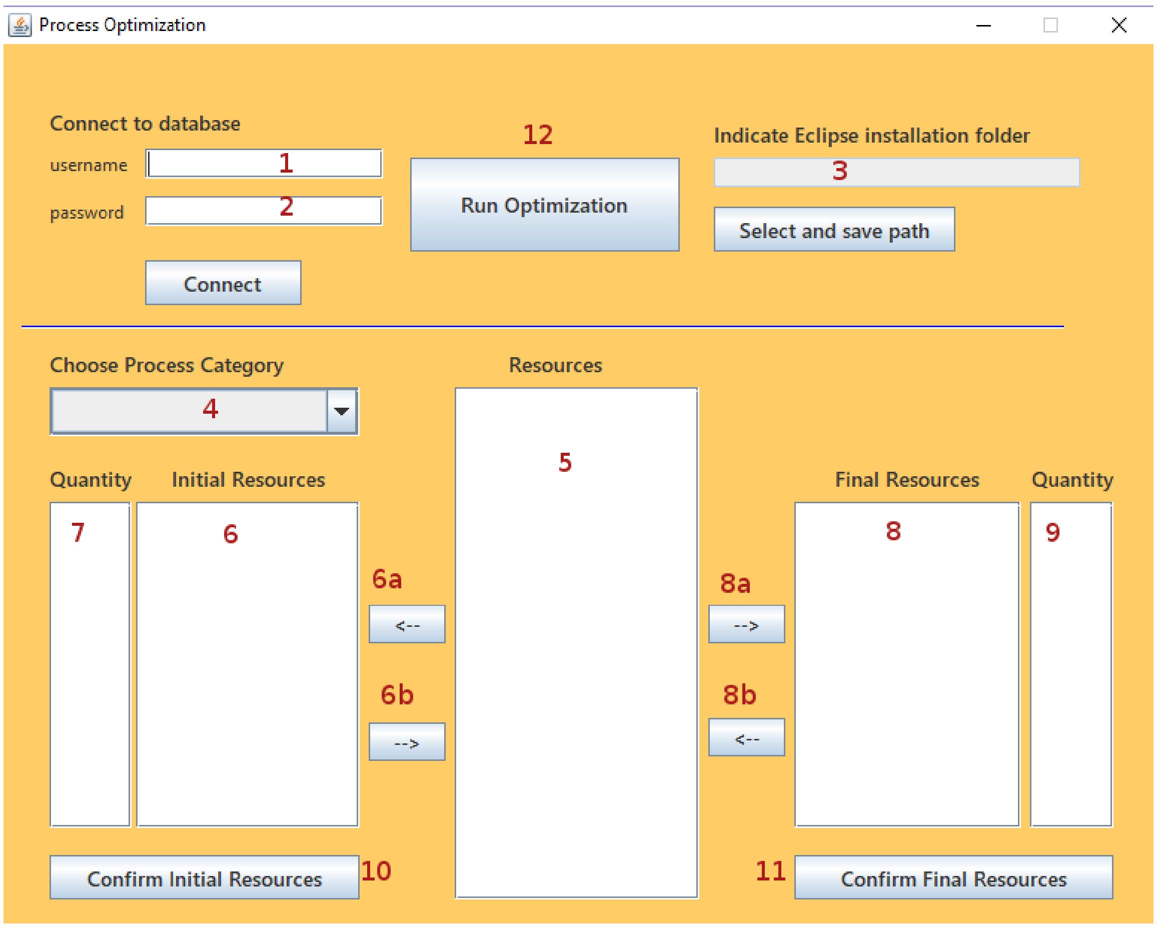


Figure 4.3.Framework initial screen

At the same time, the field *Resources* displays the available resources that are contained in the database (Figure 4.3, form 5). The user can choose the desired resources and manipulate the arrow-controls 6a and 6b to insert or remove them respectively to the set of the initial input resources (Figure 4.3, form 6). The same sequence of steps must be conducted for the selection of the final output resources (Figure 4.3, arrow-controls 8a, 8b and form 8). An additional responsibility of the user is to insert the quantities of the selected initial input and final output, utilizing the fields that are marked with the numbers 7 and 9 respectively (Figure 4.3). The termination of the data import stage is indicated by the confirmation of selections with the button-controls *Confirm Initial Resources* and *Confirm Final Resources* (Figure 4.3, button-controls 10 and 11).

The final required interaction with the user consists of the activation of the button-control *Run Optimization* (Figure 4.3, button-controls 12). The last action initiates the optimization phase of the framework's functionality. Specifically, first the data reduction operations are applied, namely the rejection of the tasks that require an input resource that is included in O_{BP} and the rejection of tasks that produce an output resource that is included in I_{BP}. A very important operation at this point is the sorting of input resources of the remaining tasks in ascending order according to their id. These tasks will become the facts for the CLP program to be constructed using the programming language Prolog. Later in this chapter we will see that the sorted input resources is a crucial characteristic for a clustering technique that is applied to the task set and reduces the computational complexity. The fully developed CLP program contains the aforementioned facts, the sets of the initial input and output sorted in ascending order according to the resources' ids and a set of constraints in the form of logical rules, in order to formulate mathematically the problem under examination and achieve the satisfaction of the objective goals.

Subsequently, the Java interface offers the communication channel for the interaction between the Java application and the ECL^{i}PS^e platform. Specifically, the CLP program is inscribed into a file, which is passed as an argument to the appropriate routines pursuing to engage the functionalities of the ECL^{i}PS^e platform that implement the main part of the optimization procedure. The inference mechanism of the ECL^{i}PS^e platform tries to determine the optimum feasible process design for the specific input and output resources. Finally, if the ECL^{i}PS^e terminates successfully, returns the result to the Java interface to be appropriately transformed for the visualization process.

4.5.1 Application's architecture

For the complete technical description of the software architecture, subsequently will be presented the necessary UML diagrams of designing level. The following *Use Case Diagram* (Figure 4.4) demonstrates graphically the structural architecture, the dependencies and the interactions between the programmatic modules that compose the framework.

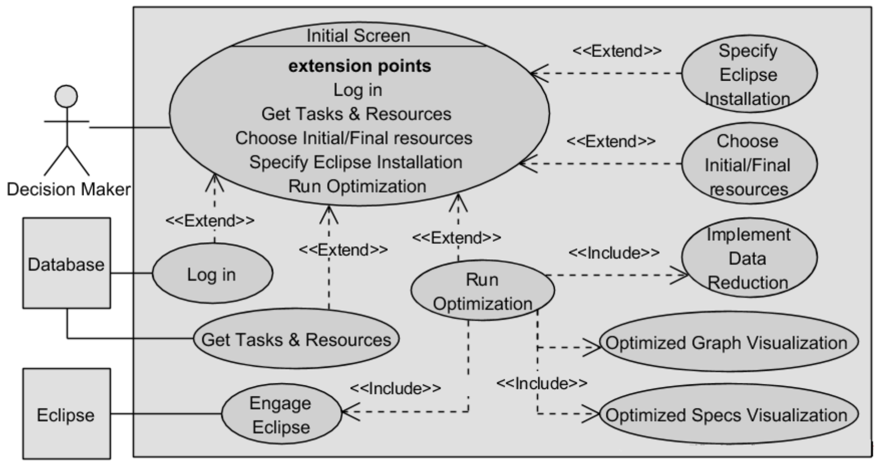


Figure 4.4.Use Case Diagram

Moreover, the *Activity Diagram* (Figure 4.5) exhibits the functional sequence of application's execution and the potential control flow paths.

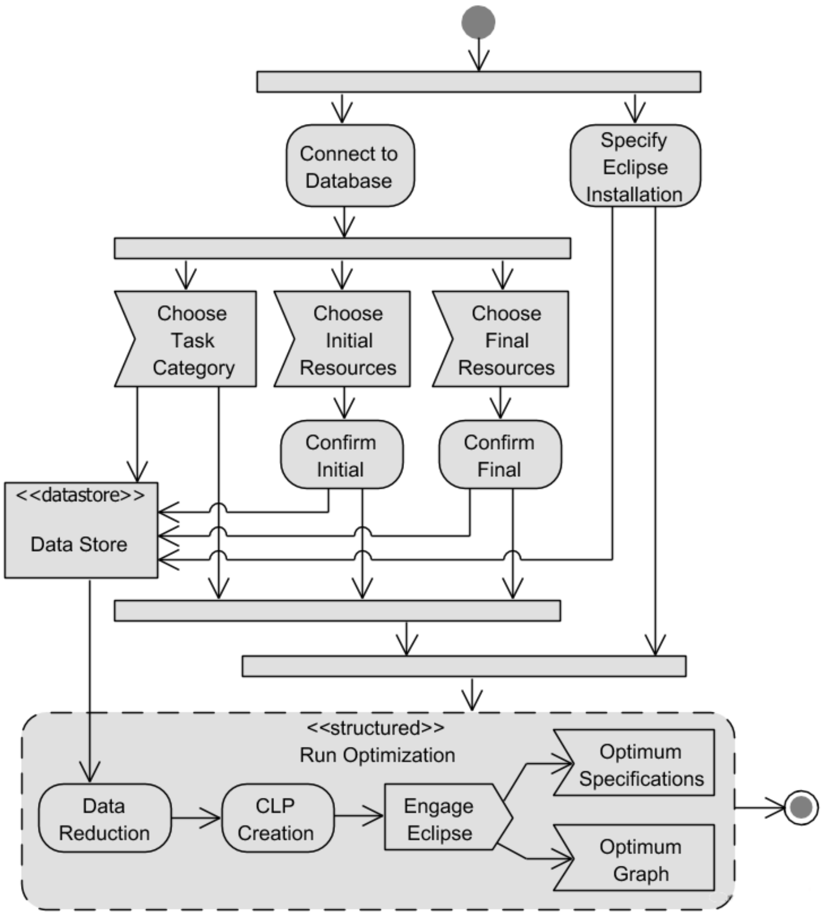


Figure 4.5.Activity Diagram

The *Sequence Diagram* (Figure 4.6) describes the consecutive foreground and background steps of a complete functional circle, for the decision of the optimized business process design for a specific case.

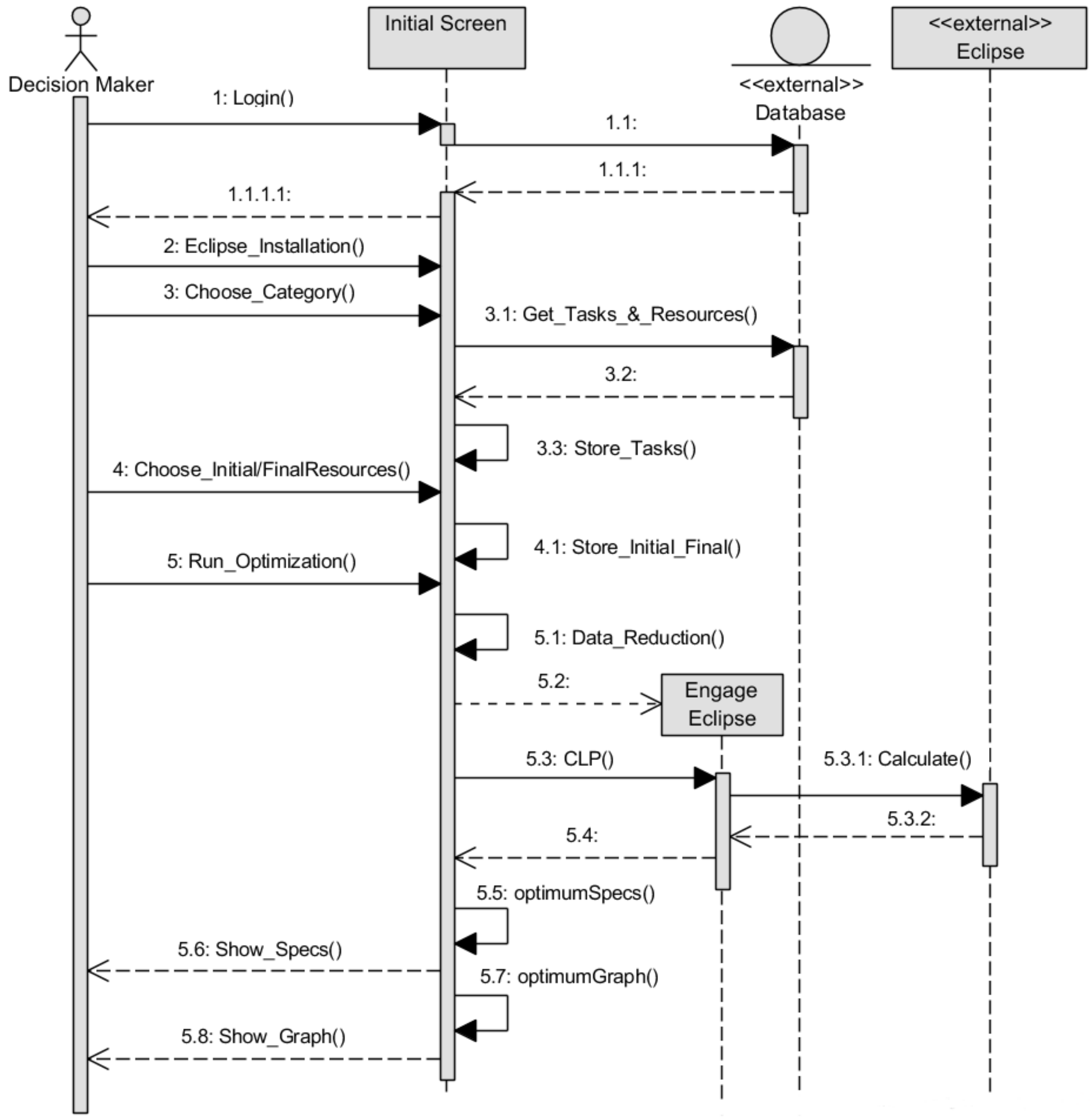


Figure 4.6.Sequence Diagram

Finally, the *Class Diagram* (Figure 4.7) shows the main structural classes of the application and the associations between them that implement the functionalities of the framework.

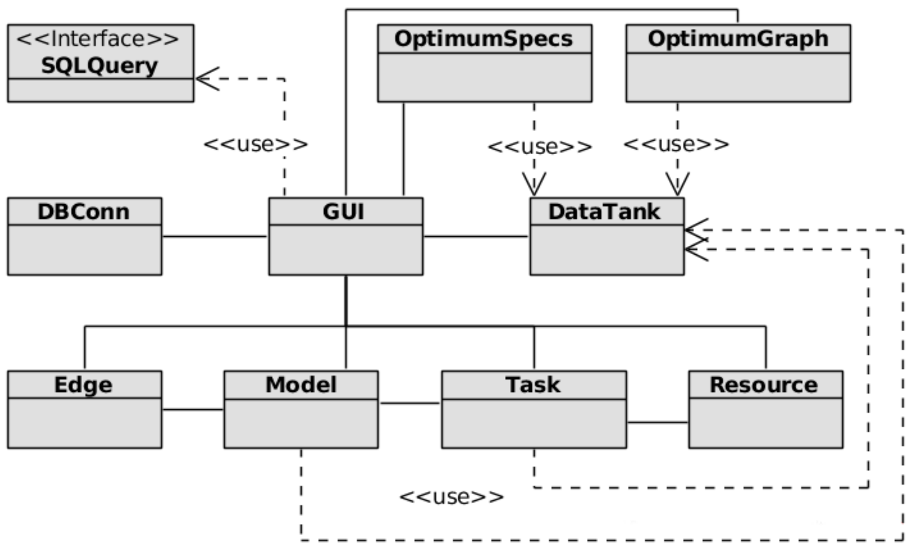


Figure 4.7.Class Diagram

4.6 Engine of inference – ECL^{i}PS^e

During mid-eighties, the European Computer-Industry Research Center in Munich initiated a research about the benefits of the application of the mathematical reasoning to practical problems (Apt and Wallace, 2007). The outcome of the aforementioned research was the development of three programming systems upon the same logic paradigm. The first for the confrontation of complex problems on multiprocessor environments, the second for the intelligent processing of vast amounts of data with advanced database techniques and the third was the programming language CHIP. The constraint logic language CHIP incorporates the concept of constraints' satisfaction with the logic programming by implementing a top-down search in order to attribute values to constraint variables from predefined finite domains (Apt and Wallace, 2007).

The merge of the three systems constitutes the ECL^{i}PS^e platform, which initially derived its constraint programming features mainly from CHIP. Over time though, ECL^{i}PS^e has been equipped with a variety of constraint solvers, solver interfaces and interfaces to Database Management Systems and other programming languages (Apt and Wallace, 2007). The last decade ECL^{i}PS^e has been used from institutions and companies all over the world for education, research business purposes in a variety of fields such as production planning, transportation scheduling, bioinformatics, software testing and network optimization (Apt and Wallace, 2007). For the purposes of the suggested framework the CLP program contains as facts the initially input resources and the required output resources sorted in ascending order according their id in the form:

```
global_input([[name, quantity],..., [name, quantity]]).
global_output ([[name, quantity],..., [name, quantity]]).
```

The available tasks are also CLP facts in the following form, having their input resources sorted as well according their id:

```
model(id(),cost(),revenue(),duration(),quality(),flexibility(),
in([[name,quantity],...,[name,quantity]]),out([[name,quantity],...,
name,quantity]], mutual_exclusions[id_a,...,id_k]).
```

The optimization procedure is conducted by nineteen predicates, which manipulate the facts in order to discover feasible process designs and decide the optimum design for the specific input and output. The first predicate is used for the construction of a feasible process design graph. Specifically, when a task *i* becomes a candidate for participation to a feasible process design, the predicate attaches to its output resources the task id, which later is used as the origin node for the edge towards the nodes that represent the tasks of the design that consume the output resources of *i.* Additionally, it keeps ascending order of the resource sets.

```
add_origin(Task_ID,Task_Res,Origin_Res):-
  add_origin_aux(Task_ID,Task_Res,[],Unsort_Origin_Res),
  msort(Unsort_Origin_Res,Origin_Res),!.

add_origin_aux(_,[],Origin_Res,Origin_Res):-!.

add_origin_aux(Task_ID,[Res|RestRes],TillNow_Res,Origin_Res):-
  append(Res,[Task_ID],Origin_H),
  append(TillNow_Res,[Origin_H],New_TillNow_Res),
  add_origin_aux(Task_ID,RestRes,New_TillNow_Res,Origin_Res).
```

The second predicate applies a type of clustering to the tasks set according the id of their first input resource, creating twenty groups using the predefined bounds as separation criteria. The bounds are determined by the Java interface in the base of the tasks set size, trying to create task groups of the same size. Aim of this technique is the reduction of searching space in each iteration of the algorithm by excluding the majority of the tasks that cannot participate to the process design with criterion the currently available resources. The key point is to keep always the input resources sets sorted in ascending order according their id. Each of the task groups contains task with the characteristic that the id value of their first input resource is within a specific range which is determined by the given bounds.

In each iteration, the algorithm selects only the task groups with the resource id range where belong the ids of the currently available input resources and ignores the others. For at least one of the currently available input resources, if there is a feasible process design then there is a task in the corresponding task group that its first input resource have the same id. If there is not such a task for any of currently available input resources then there is not a feasible design and the algorithm. Additionally, when is checked the appropriateness of a task to participate to a feasible design and its first input resource is not contained in the currently available input resources then it can be rejected without checking its other input resources.

```
make_task_groups([Task_group1,Task_group2,Task_group3,Task_group4,Task_
group5,Task_group6,Task_group7,Task_group8,Task_group9,Task_group10,
Task_group11,Task_group12,Task_group13,Task_group14,Task_group15,
Task_group16,Task_group17,Task_group18,Task_group19,Task_group20]):-

  bounds([B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,B1
  8,B19]),

  findall(T1,(task(id(T1),_,_,_,_,_,in([[H1|_]|_]),_,_),H1 =< B1),
  Task_group1),
  findall(T2,(task(id(T2),_,_,_,_,_,in([[H2|_]|_]),_,_),H2 > B1,H2 =<
  B2),Task_group2),
  findall(T3,(task(id(T3),_,_,_,_,_,in([[H3|_]|_]),_,_),H3 > B2,H3 =<
  B3),Task_group3),
  findall(T4,(task(id(T4),_,_,_,_,_,in([[H4|_]|_]),_,_),H4 > B3,H4 =<
  B4),Task_group4),
  findall(T5,(task(id(T5),_,_,_,_,_,in([[H5|_]|_]),_,_),H5 > B4,H5 =<
  B5),Task_group5),
  findall(T6,(task(id(T6),_,_,_,_,_,in([[H6|_]|_]),_,_),H6 > B5,H6 =<
  B6),Task_group6),
  findall(T7,(task(id(T7),_,_,_,_,_,in([[H7|_]|_]),_,_),H7 > B6,H7 =<
  B7),Task_group7),
  findall(T8,(task(id(T8),_,_,_,_,_,in([[H8|_]|_]),_,_),H8 > B7,H8 =<
  B8),Task_group8),
  findall(T9,(task(id(T9),_,_,_,_,_,in([[H9|_]|_]),_,_),H9 > B8,H9 =<
  B9),Task_group9),
  findall(T10,(task(id(T10),_,_,_,_,_,in([[H10|_]|_]),_,_),H10 > B9,H10
  =< B10),Task_group10),
  findall(T11,task(id(T11),_,_,_,_,_,in([[H11|_]|_]),_,_),H11 > B10,
  H11 =< B11),Task_group11),
  findall(T12,(task(id(T12),_,_,_,_,_,in([[H12|_]|_]),_,_),H12 > B11,
  H12 =< B12),Task_group12),
  findall(T13,(task(id(T13),_,_,_,_,_,in([[H13|_]|_]),_,_),H13 > B12,
  H13 =< B13),Task_group13),
  findall(T14,(task(id(T14),_,_,_,_,_,in([[H14|_]|_]),_,_),H14 > B13,
  H14 =< B14),Task_group14),
  findall(T15,(task(id(T15),_,_,_,_,_,in([[H15|_]|_]),_,_),H15 > B14,
  H15 =< B15),Task_group15),
  findall(T16,(task(id(T16),_,_,_,_,_,in([[H16|_]|_]),_,_),H16 > B15,
  H16 =< B16),Task_group16),
  findall(T17,(task(id(T17),_,_,_,_,_,in([[H17|_]|_]),_,_),H17 > B16,
  H17 =< B17),Task_group17),
  findall(T18,(task(id(T18),_,_,_,_,_,in([[H18|_]|_]),_,_),H18 > B17,
  H18 =< B18),Task_group18),
  findall(T19,(task(id(T19),_,_,_,_,_,in([[H19|_]|_]),_,_),H19 > B18,
  H19 =< B19),Task_group19),
  findall(T20,(task(id(T20),_,_,_,_,_,in([[H20|_]|_]),_,_),H20 > B19),
  Task_group20).
```

The third predicate applies binary search in each iteration of the algorithm, in order to find the task groups with resource id range that corresponds to the ids of the currently available input resources.

```
select_group(Task_groups,Global_input,Group):-
  bounds([B1,B2,B3,B4,B5,B6,B7,B8,B9,B10, B11,B12,B13,B14,B15,B16,B17,
  B18,B19]),
  member([Id|_],Global_input),

   (Id =< B10 ->
     (Id =< B5 ->
       (Id =< B3 ->
         (Id =< B1 ->
           (nth1(1,Task_groups,Group))
         ;
           (Id =< B2 ->
             (nth1(2,Task_groups,Group))
           ;
             (nth1(3,Task_groups,Group))
           )
         )
       ;
         (Id =< B4 ->
           (nth1(4,Task_groups,Group))
         ;
           (nth1(5,Task_groups,Group))
         )
       )
     ;
       (Id =< B7 ->
         (Id =< B6 ->
           (nth1(6,Task_groups,Group))
         ;
           (nth1(7,Task_groups,Group))
         )
       ;
         (Id =< B9 ->
           (Id =< B8 ->
             (nth1(8,Task_groups,Group))
           ;
             (nth1(9,Task_groups,Group))
           )
         ;
```

```
          (nth1(10,Task_groups,Group))
        )
      )
    )
  ;
    (Id > B15 ->
      (Id > B17 ->
        (Id > B19 ->
          (nth1(20,Task_groups,Group))
        ;
          (Id > B18 ->
            (nth1(19,Task_groups,Group))
          ;
            (nth1(18,Task_groups,Group))
          )
        )
      ;
        (Id > B16 ->
          (nth1(17,Task_groups,Group))
        ;
          (nth1(16,Task_groups,Group))
        )
      )
    ;
      (Id > B13 ->
        (Id > B14 ->
          (nth1(15,Task_groups,Group))
        ;
          (nth1(14,Task_groups,Group))
        )
      ;
        (Id > B12 ->
          (nth1(13,Task_groups,Group))
        ;
          (Id > B11 ->
            (nth1(12,Task_groups,Group))
          ;
            (nth1(11,Task_groups,Group))
          )
        )
      )
    )
  ).
```

The fourth predicate removes for the rest of the search procedure the tasks that have already been selected to participate to a feasible process design and ensures the satisfaction of the logic formula 4.

```
replace_group(Task_groups,Group,New_Group,New_Task_groups):-
  append(Start,[Group|Fin],Task_groups),
  append(Start,[New_Group|Fin],New_Task_groups),!.
```

The fifth predicate checks if the input resources of a task are contained into the set of currently available resources in equal or bigger quantity. In case of success, the task becomes a candidate for participation to the process design, the quantities of its input resources are subtracted from the quantities of the corresponding available resources and it is excluded from the next steps of the optimization procedure. Finally, the new currently available input is sorted in ascending order according to the resources' ids. To make the connection with the mathematical formulation of the problem, this predicate checks the satisfaction of the logic formula 1.

```
check_input(_,[],Global_input,Global_input,[]):-!.

check_input(Task_ID,[[Task_resource,Task_quantity]|Task_input],
Global_input,New_Global_input,[(Origin_ID,Task_ID)|RestConn]):-
  member([Task_resource,Global_quantity,Origin_ID],Global_input),
  arrange_resource_consumption(Task_quantity,Global_quantity,
  Task_resource,Origin_ID,Task_input,New_Task_Input,Global_input,
  Temp_Global_Input),
  check_input(Task_ID,New_Task_Input,Temp_Global_Input,
  New_Global_input, RestConn).

arrange_resource_consumption(Quantity,Quantity,Task_resource,
Origin_ID,Task_input,Task_input,Global_input,New_Global_input):-
  !,
  lists:delete([Task_resource,Quantity,Origin_ID],Global_input,
  New_Global_input).

arrange_resource_consumption(Task_quantity,Global_quantity,
Task_resource,Origin_ID,Task_input,Task_input,Global_input,
New_Global_input):-
  Task_quantity #=< Global_quantity,!,
  New_Global_quantity is Global_quantity - Task_quantity,
  lists:delete([Task_resource,Global_quantity,Origin_ID],Global_input,
  Temp_Global_input),
  msort([[Task_resource,New_Global_quantity,Origin_ID]|
  Temp_Global_input],New_Global_input).

arrange_resource_consumption(Task_quantity,Global_quantity,
Task_resource,Origin_ID,Task_input,New_Task_input,Global_input,
New_Global_input):-
  Task_quantity #> Global_quantity,!,
  New_Task_quantity is Task_quantity - Global_quantity,
  lists:delete([Task_resource,Global_quantity,Origin_ID],
  Global_input,New_Global_input),
  lists:delete([Task_resource,Task_quantity],Task_input,
  Temp_Task_input),
  msort([[Task_resource,New_Task_quantity,Origin_ID]|
  Temp_Task_input],New_Task_input).
```

The sixth predicate checks if some of the currently available input resources belong to the set of required (global) output and cover them quantitatively in total or partially. When succeeds, the quantities of the available input resources are subtracted from the quantities of the corresponding required output resources but also they are removed from the available input resources because they must not be used as input resources by any other task as the optimization procedure continues. In addition, it sorts the remaining final output in ascending order. This predicate refers to the satisfaction of the logic formulas 2 and 5.

```
check_output(Global_input,Global_output,New_Global_input,New_Global_out
put,[(Origin_ID,-2)|RestConn]):-
  member([Output_resource,Output_quantity],Global_output),
  member([Input_resource,Input_quantity,Origin_ID],Global_input),
  Input_resource #= Output_resource,
  Input_quantity #=< Output_quantity,
  New_Output_quantity #= Output_quantity - Input_quantity,
  lists:delete([Output_resource,Output_quantity],Global_output,
  Global_output_aux),
  (New_Output_quantity #= 0 ->
  Temp_Global_output = Global_output_aux;
  append([[Output_resource,New_Output_quantity]],Global_output_aux,
  Temp_Global_output)),
  lists:delete([Input_resource,Input_quantity,Origin_ID],
  Global_input,Temp_Global_input),
  check_output(Temp_Global_input,Temp_Global_output,New_Global_input,
  Unsort_Global_output,RestConn),
  msort(Unsort_Global_output,New_Global_output),!.

check_output(Global_input,Global_output,Global_input,Global_output,_):-
  !.
```

The functionality of the seventh predicate is rather procedural. In particular, when a task becomes candidate for participation to a feasible design, adds its output resources to the set of the currently available input resources for the next step of the optimization procedure, participating to the check of satisfaction of the logic formula 3. Additionally, it maintains the ascending order of the currently available input

```
add_task_output([],Global_input,Global_input):-!.

add_task_output([[Output_resource,Output_quantity,Origin_ID]|
Rest_Task_output],Global_input,New_Global_input):-
  append([[Output_resource,Output_quantity,Origin_ID]],Global_input,
  Temp_Global_input),
  add_task_output(Rest_Task_output,Temp_Global_input,
  Unsort_Global_input),
  msort(Unsort_Global_input,New_Global_input).
```

The eighth predicate could be considered as the heart of the optimization procedure. Specifically, using the aforementioned predicates decides whether a task can be carried out with the currently available input resources and thus, can be a candidate for participation to a feasible process design.

```
participation(Task_group,Global_input,Global_output,Task_ID,
Cost,Revenue,Quality,Flexibility,Rest_tasks,New_Global_input,
New_Global_output,[Task_Conns1|Task_Conns2]):-
  member(Task_ID,Task_group),
  task(id(Task_ID),cost(Cost),revenue(Revenue),_,quality(Quality),
  flexibility(Flexibility),in(Task_input),out(Task_output),_),
  check_input(Task_ID,Task_input,Global_input,Temp_Global_input,
  Task_Conns1),
  add_origin(Task_ID,Task_output,Origin_Task_output),
  add_task_output(Origin_Task_output,Temp_Global_input,
  Temp_Global_input1),
  check_output(Temp_Global_input1,Global_output,New_Global_input,
  New_Global_output,Task_Conns2),
  lists:delete(Task_ID,Task_group,Rest_tasks).
```

The ninth predicate applies the searching and scheduling constraints to the set of the available task and constructs a feasible process design, taking under consideration the specific initially input resources and the specific initially required output resources. This predicate succeeds when the termination conditions 1, 2 and 3 have been satisfied.

```
design(Model,Tcost,Trevenue,Tduration,Tquality,Tflexibility,
TEvaluation,Connectivity,Starts,Fins):-
  make_task_groups(Task_groups),
  global_input(Global_input),
  add_origin(-1,Global_input,Origin_GI),
  global_output(Global_output),
  design_aux(Task_groups,Origin_GI,Global_output,Model,Tcost,Trevenue,
  TQuality,TFlexibility,Conns),
  flatten(Conns,Connections),
  remove_duplicates(Connections,Connectivity),
  model_durations_exclusions(Model,Durations,Mutual_Exclusions),
  timespan(Model,Connectivity,Durations,Mutual_Exclusions,Starts,
  Fins,TDuration),
  Evaluation #= TRevenue - TCost - TDuration + TQuality + TFlexibility,
  labeling([TRevenue,TCost,TDuration,TQuality,TFlexibility]),
  TEvaluation is -Evaluation.

design_aux(_,[],[],[],0,0,1000,1000,_):-!.

design_aux(Task_groups,Global_input,Global_output,[Task_ID|
RestModel],TCost,TRevenue,TQuality,TFlexibility,[Task_Conns|
TillNow_Conn]):-
  setof(Group,select_group(Task_groups,Global_input,Group),Sub_Groups),
  member(Sub_Group,Sub_Groups),
  participation(Sub_Group,Global_input,Global_output,Task_ID,
  Cost1,Revenue1,Quality1,Flexibility1,New_Sub_Group,New_Global_input,
  New_Global_output,Task_Conns),
  replace_group(Task_groups,Sub_Group,New_Sub_Group,New_Task_groups),
  design_aux(New_Task_groups,New_Global_input,New_Global_output,
  RestModel,Cost2,Revenue2,Quality2,Flexibility2,TillNow_Conn),
  TCost #= Cost1 + Cost2,
  TRevenue #= Revenue1 + Revenue2,
  TQuality #= min(Quality1,Quality2),
  TFlexibility #= min(Flexibility1,Flexibility2).
```

The tenth predicate is utilized for right the visualization of a feasible process design graph, replacing the recurring edges with just one.

```
remove_duplicates(List,UList):-
  remove_duplicates_aux(List,[],UList).

remove_duplicates_aux([],UList,UList):-!.

remove_duplicates_aux([H|Rest],TempUList,UList):-
  member(H,TempUList),
  remove_duplicates_aux(Rest,TempUList,UList),!.

remove_duplicates_aux([H|Rest],TempUList,UList):-
  remove_duplicates_aux(Rest,[H|TempUList],Ulist),!.
```

The eleventh predicate is auxiliary and is used to find the duration and the lists of mutual exclusions of the tasks that participate to a feasible process design.

```
model_durations_exclusions([],[],[]).


model_durations_exclusions([ID|Model],[Dur|Durations],
New_Mutual_Exclusions):-
  task(id(ID),_,_,duration(Dur),_,_,_,_,mutual_exclusions(ME_IDs)),
  model_durations_exclusions(Model,Durations,Mutual_Exclusions),
  fix_arg_ex(ME_IDs,ID,Mutual_Exclusions,New_Mutual_Exclusions).

fix_arg_ex([],_,Mutual_Exclusions,Mutual_Exclusions):-!.

fix_arg_ex(ME_Ids,ID,Mutual_Exclusions,[[ID|ME_Ids]|
Mutual_Exclusions]).
```

The twelfth predicate is responsible for the scheduling of the tasks of the feasible process design in a way that minimizes the total duration. In addition, it determines the relative start and finish time of each task, contributing to the overall optimization procedure.

```
timespan(Model_Ids,Connectivity,Durations,Mutual_Exclusions,Starts,
Fins,TDuration):-
  length(Durations,N),
  length(Starts,N),
  length(Fins,N),
  ic_global:sumlist(Durations,Duration_upper_bound),
  Starts #:: 0..Duration_upper_bound,
  Fins #:: 0..Duration_upper_bound,
  vars_matching(Starts,Fins,Durations),
  precedence_constraints(Model_IDs,Connectivity,Starts,Fins),
  exclusion_constraints(Model_IDs,Durations,Mutual_Exclusions,Starts),
  ic_global:maxlist(Fins,TDuration),
  bb_min(labeling(Starts),Tduration,bb_options
  {report_success:true/0}),!.
```

The thirteenth, fourteenth and fifteenth predicates are auxiliary and are used by the twelfth predicate for the scheduling procedure. In detail, the thirteenth predicate matches the variables that represent the start and time of the tasks of a feasible process design. The fourteenth predicate constructs the constraints that determine the precedence relations between the tasks of a feasible design and guards the satisfaction of the satisfaction of the logic formula 6, 8, 9 and 10. Finally, the fifteenth predicate constructs the constraints that define the mutual exclusion relations between the tasks of a feasible design and ensures the satisfaction of the logic formula 7.

```
vars_matching([],[],[]):-!.

vars_matching([Start|RestStarts],[Fin|RestFins],[Duration|
RestDurations]):-
  Fin #= Start + Duration,
  vars_matching(RestStarts,RestFins,RestDurations).

precedence_constraints(_,[],_,_):-!.

precedence_constraints(Model_IDs,[(From,To)|RestConns],Starts,Fins):-
  ((From #> 0,To #> 0) ->
  (nth1(IndexTo,Model_IDs,To),
  nth1(IndexFrom,Model_IDs,From),
  nth1(IndexTo,Starts,S),
  nth1(IndexFrom,Fins,F),
  S #>= F);
  true),
  precedence_constraints(Model_Ids,RestConns,Starts,Fins).

exclusion_constraints(_,_,[],_):-!.

exclusion_constraints(Model_IDs,Durations,[ME|RestME],Starts):-
  starts_list(Model_IDs,Starts,ME,Starts_List),
  durations_list(Model_IDs,Durations,ME,Durations_List),
  disjunctive(Starts_List,Durations_List),
  exclusion_constraints(Model_IDs,Durations,RestME,Starts).
```

The sixteenth, seventeenth and eighteenth predicates are used as auxiliary by the fifteenth predicate during the construction mutual exclusion constraints.

```
starts_list(Model_IDs,Starts,Excluded_List,Starts_List):-
  create_list(Model_Ids,Starts,Excluded_List,Starts_List).

durations_list(Model_IDs,Durations,Excluded_List,Durations_List):-
  create_list(Model_Ids,Durations,Excluded_List,Durations_List).

create_list(_,_,[],[]).

create_list(Model_IDs,Attributes,[EX|RestExcluded],[Attribute_EX|
Attributes_List]):-
```

```
  nth1(IndexEX,Model_IDs,EX),
  nth1(IndexEX,Attributes,Attribute_EX),
  create_list(Model_IDs,Attributes,[EX|RestExcluded],Attributes_List).
```

Finally, the nineteenth predicate implements the branch_and_bound optimization strategy trying to constructs feasible process designs and determine the design that maximizes the objective function $F(opt)$. The predicate succeeds when at least one feasible process design has been found within a time period of 60s cpu.

```
optimization(Optimum,Cost,Revenue,Duration,Quality,Flexibility,
Evaluation,Connectivity,Starts,Fins):-
  bb_min(design(Optimum,Cost,Revenue,Duration,Quality,Flexibility,
  Eval,Connectivity,Starts,Fins),
  Eval,
  bb_options{timeout:60}),
  Evaluation is -Eval.
```

4.7 Visualization

After the execution of the logic program the ECL^{i}PS^{e} platform on success returns attributes of the optimized business process design in the form: *(Tasks List, cost, revenue, duration, quality, flexibility, evaluation, Starts List, Finns List, Connectivity)*. The Java module formats the characteristics of the optimum design and prints them on the screen. Subsequently, depicts into a separate window the graph of the optimum design. For the construction of the graphs is used the free Java library JGraphT release 1.2, which provides graph theory mathematical objects and graph-related algorithms implementations (Naveh and Contributors, 2018). The JGraphT project is consisted by two complementary and collaborating libraries. The first is the JGraphT that focuses on data structures and algorithms implementation. The second is the JGraphX that is used for data rendering and GUI-based editing (Naveh and Contributors, 2018).

4.8 Asymptotic analysis

The asymptotic analysis of an algorithm refers to the determination of its computational complexity in terms of time or space requirement in relation to the size of the input. The analysis focuses on the behavior of the algorithm in the worst case, namely the growth rate of complexity as the input's size tends to infinity. The algorithm of the process optimization procedure uses the exhaustive search strategy and examines all the permutations of the set of the chosen tasks in order to determine the business process design with the optimized evaluation. Therefore, in the general case for input of size $n \to \infty$ there are $n!$ possible permutations and the computational complexity can be estimated by the approximation of its natural logarithm $\ln n! = \sum_{x=1}^{n} \ln x$ (Wikipedia contributors, 2018). Specifically:

$$\int_{1}^{n} \ln x \, dx \le \sum_{x=1}^{n} \ln x \le \int_{0}^{n} \ln(x+1) \, dx \approx$$

$$n \ln\left(\frac{n}{e}\right) \le \ln n! \le (n+1) \ln\left(\frac{(n+1)}{e}\right) + 1 \approx$$

$$\left(\frac{n}{e}\right)^{n} e \le n! \le \left(\frac{n+1}{e}\right)^{n+1} \cdot e \approx$$

$$\frac{n^{n}}{3} \le n! \le \frac{n^{n}}{2}$$

The asymptotic analysis shows that the proposed algorithm has exponential complexity:

$$O\left(\left(\frac{n}{2}\right)^{n}\right)$$

and in the general case, it is not computationally efficient. However, the framework applies three data reduction techniques, which can lead to a significant reduction of the searching domain and improve the overall performance and thus, the computational efficiency of the algorithm. More specific:

1. The hierarchical indexing schema of the task library restricts the search only to a category of tasks and in practice the number of tasks that belong to a specific functional sector of a company according to Porter's Value Chain is limited. Furthermore, a systematic analysis of a company can lead to a further refinement of its operational structure and the definition of more functional sub-sectors that contain small number of tasks.
2. The rejection of the tasks that require an input resource that is included in O_{BP} or produce an output resource that is included in I_{BP} can diminish further the size of the task set that is used as input to the algorithm.
3. The clustering of the input task set into twenty groups and the selection of a small number of these groups in each iteration of the algorithm, decreases significantly the searching domain.

4.9 Summary

The fourth chapter presented the rationale that used for the design and construction of the proposed framework. Initially, is analyzes in mathematical notation the formulation of the problem under examination. Next, it presents the modules that consist the framework both in terms of structure and functionality. Specifically, the MySQL database contains all the necessary data while the Java module is responsible for the interconnection between the subsystems and the interaction with the user. The ECL^{i}PS^e platform conducts the main part of the optimization procedure and finally, the JGraphT project visualizes obtained result. Although the algorithm typically has exponential computational complexity, in practice the indexing schema of the task library, the rejection of tasks with specific characteristics and the clustering of the input task set, reduce significantly the searching domain and improves satisfactorily the performance.

5 Task generator

The present chapter describes the design and implementation of a generator for the construction of instances for the business process construction and scheduling problem, as it have been formulated in the context of the present thesis. The generated instances can be of various levels of difficulty, depending on the user's input and the leverage of three complexity factors that correspond to three major structural characteristics of a business process, if the latter is considered as an acyclic directed graph. More specifically, the Net Complexity factor declares the maximum number of precedence relations that a task can participate in, the Resource Density factor determines the number of input and output resources that a task can consume and produce respectively and finally, the Inter-level

Connection factor denotes if the immediate successors of a task belong to the same level of the business process tree-graph or that is possible to exists precedence relations between tasks of any level. Additionally, the difficulty level of an instance can be increased by the ability of the generator to create alternative designs in the base of one initially constructed feasible process design. In general, an instance can be described as a set of a predefined number of tasks, which contains at least two feasible business process designs with respect to a certain set of initial input resources and a certain set of required final output resources. The creation of a set of instances of diverse levels of complexity and size, provides the tool for the systematic evaluation of the computational complexity and the performance of the business process optimization algorithm.

5.1 Notation and task representation

We consider that a business process design is consisted of $N + 2$ labeled and partially ordered tasks, where the $n = 0$ task is the unique dummy start-node and the n = $N + 1$ is the unique dummy end-node. In order to constitute a processes, each task consumes a set of specific input resources and produces a set of specific output resources. Each task is characterized by a vector of five measurable attributes, namely cost, revenue, duration, quality and flexibility. The evaluations for these attributes are generated randomly within certain lower and upper bounds set by the user (arguments to the algorithm). A generated instance of the problem contains tasks that can be distinguished into two types. The first type refers to at least one subset of biased tasks, which regard to the initial input, the final output and the individual input/output resources, form a feasible business process design. The second type is the sub-set of arbitrary tasks, which serve only as a control of the instance's size and cannot participate to the construction of a feasible process design. Table 5.1 presents the utilized notations for the user input and provides their definition.

Table 5.1.Notation and definitions

User Input	**Definition**
process_size	The number N of tasks that construct the business process.
population	The number of arbitrary tasks that are contained in the problem instance.
min_task_id *max_task_id*	The range of integers for the IDs of the tasks.
min_cost *max_cost*	The lower and upper bound of the randomly calculated task's cost respectively.
revenue_margin	The percentage of profit that is produced by a task.
min_duration *max_duration*	The lower and upper bound of the randomly calculated task's duration respectively.
quality_scale	The upper bound of integers for the evaluation of the quality of a task (1...N).
flexibility_scale	The upper bound of integers for the evaluation of the flexibility of a task (1...M).
min_resource_id *max_resource_id*	The range of integers for the IDs of the resources.
min_resource_quantity *max_resource_quantity*	The range within the quantity of an input / output resource must be.
global_input_ids *global_input_quantities*	The IDs and the corresponding quantities of the initially available input resources.
global_output_ids *global_output_quantities*	The IDs and the corresponding quantities of the required output resources by the constructed business process.
net_complexity	The integer value of the network complexity factor.
resource_factor	The integer value of the resource factor.
inter_connection	The boolean value of the inter_connection factor.
mutual_exclusions(task_id: $id_k,\dots,id_n$*)*	For each task, the list of tasks (if there are) that cannot be executed in concurrently.

For the manipulation of the complexity level of a problem's instance, the generator utilizes three parameters, namely the Network complexity factor, the Resource Density factor and the Inter-level Connection factor. Furthermore, the user can choose the construction of a number of alternative feasible designs in the base of one initially constructed by the generator. In detail:

1. **Network Complexity factor**: the complexity of the business process network can be measured by the mean number of unique connections (precedence relations) per node (task). The rationale behind this parameter is that for certain number of tasks, the higher the complexity is, the more dependencies there are between the tasks, affecting their scheduling and the makespan of the entire business process (Kolisch, Sprecher and Drexl, 1995; Kolisch, Schwindt and Sprecher, 1999). According to graph theory, a complete graph is a graph where all nodes are connected with each other, thus for a graph of *n* nodes, each node is connected with the *(n – 1)* other nodes but in total there are *n * (n – 1) / 2* unique directed arcs excluding the duplicates. Considering that for realistic business processes each task can be related up to 60% of the other tasks, the Network Complexity factor takes values from 1 to 6 and randomly determines the percentage of tasks which each task is related to as shown in Table 5.2:

Table 5.2.Network Complexity

Network complexity	net_complexity value
1	*(random in [1, 10]) / 100* relations with 1% to 10% of the remaining tasks
2	*(random in [11, 20]) / 100* relations with 11% to 20% of the remaining tasks
3	*(random in [21, 30]) / 100* relations with 21% to 30% of the remaining tasks
4	*(random in [31, 40]) / 100* relations with 31% to 40% of the remaining tasks
5	*(random in [41, 50]) / 100* relations with 41% to 50% of the remaining tasks
6	*(random in [51, 60]) / 100* relations with 51% to 60% of the remaining tasks

In order to ensure that the resulting relation graph is a tree, and also ensure the connectivity of the business process design, for each task the number of its precedence relations is calculated by the formula:

*connections_number = max(1, round((process_size - 1) * net_complexity))* *(1)*

where *process_size* initially is given by the user and is decreased by one each time a task has been processed and has acquired precedence relations.

2. **Resource Density factor**: determines the number of different resources that a task produces as output, thus decides the number of the different resource that are input

of the successive tasks of each processed task. This increases the level of dependency among the tasks that participate to the construction of the process design and can affect their execution time scheduling and the search complexity of the optimization algorithm. In order to insert randomness, the Resource Density factor is utilized as presented in Table 5.3:

Table 5.3.Resource Density

Resource Density value	Output resource number
1	*1*
2	*random in [1, 3]*
3	*random in [2, 4]*
4	*random in [3, 5]*
5	*random in [4, 6]*
6	*random in [5, 7]*

3. **Inter-level Connection factor**: it is a boolean parameter that determines if the immediate successors of a task in terms of resources consumption belong to the same level of the business process tree-graph or the precedence relations can exist regardless of the level, creating groups of dependency. If the factor is set to "false" it means that the resources produced by a task are consumed by the directly successive tasks, otherwise the resources can be consumed by tasks that are in a greater depth of the tree hierarchy. This kind of organization increases the expansion and diffusion dependency between the nodes of a graph and may create more backtracking points for the optimization algorithm. The following graphs depict more clearly the differences in a business process tree-graph structure accordingly to the value of the Inter-level Connection factor. In detail, we used the same sets of initial input and final output resources for the construction of a feasible business process design of 10 and 15 tasks. Figure 5.1 shows the graph of the 10 task design with the Inter-level Connection factor set to false (0), while Figure 5.2 shows the graph of the 10 task design with the Inter-level Connection factor set to true (1).

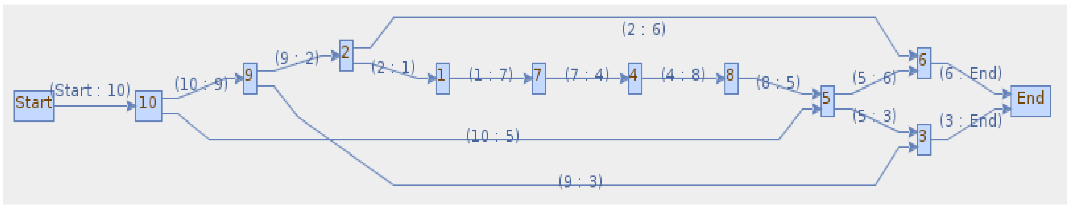


Figure 5.1: 10 tasks design with ICF = false (0)

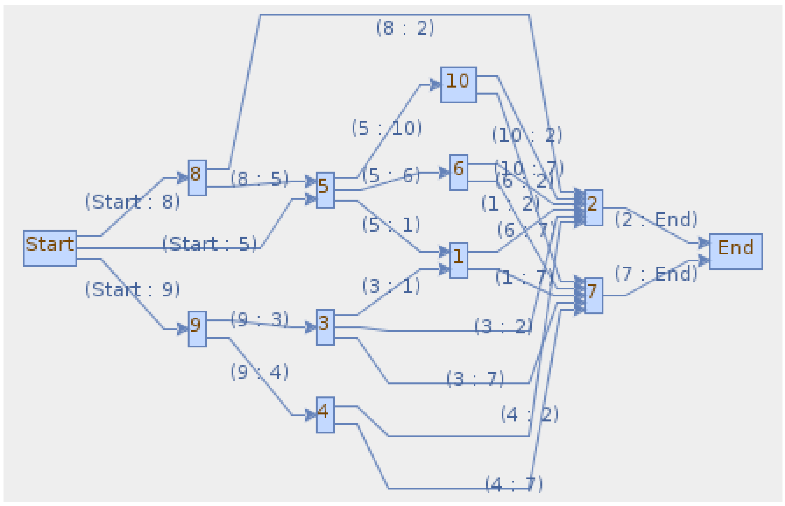


Figure 5.2: 10 tasks design with ICF = true (1)

Similarly, Figure 5.3 shows the graph of the 15 task design with the Inter-level Connection factor set to false (0), while Figure 5.4 shows the graph of the 10 task design with the Inter-level Connection factor set to true (1).

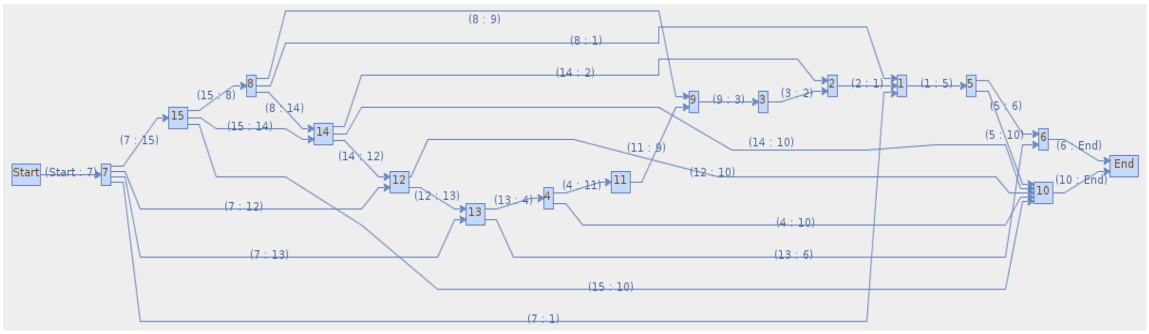


Figure 5.3: 15 tasks design with ICF = false (0)

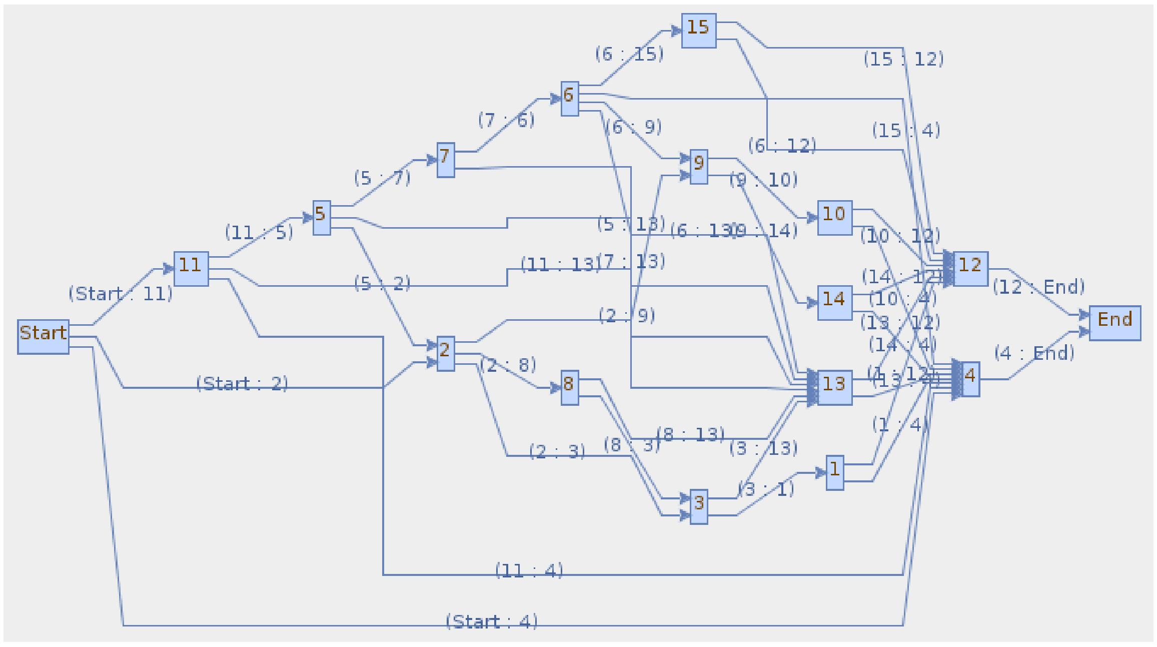


Figure 5.4: 15 tasks design with ICF = true (1)

4. **Alternative feasible designs**: the creation of alternative designs in the base of one initially constructed feasible process design is based mathematically on the principles of Combinatorics. Specifically, considering a process design of n tasks as a vector of variables then there are $k_1 * k_2 * \ldots * k_n$ possible evaluations, where k_i is the number of different values that can take the variable i. Therefore, if $x < n$ task of a constructed process design is duplicated then there are 2^x alternative designs. The generator provides the choice for the creation of 2, 4, 8 or 16 alternatives.

Finally, for the representation of the tasks is used a form that makes the generated instances structurally compatible as input to the business process optimization algorithm, that is:

```
task(id, cost, revenue, duration, quality, flexibility, input(...),
output(...), mutual_exclusions[mutual_exclusions])
```

5.2 Task generation algorithm

The instance generator algorithm consists of two subroutines, one for the biased task set calculation and one for the arbitrary task set calculation respectively. The subroutines can operate independently or simultaneously, depending on the user input for `process_size` and `population` variables. When `process_size > 0` is triggered the biased task set calculation and the arbitrary set task calculation is triggered when `population > 0`. First of all the algorithm creates the data structures needed by both subroutines, that is the initial input, the final output and the set of resources that will be used as the input/output of the tasks:

```
READ
  process_size
  population
  min_task_id, max_task_id
  min_cost, max_cost
  revenue_margin
  min_duration, max_duration
  quality_scale, flexibility_scale
  min_resource_id, max_resource_id
  min_resource_quantity, max_resource_quantity
  global_input_ids, global_input_quantities
  global_output_ids, global_output_quantities
  net_complexity
  resource_factor
  inter_connection
  mutual_exclusions
END READ

BEGIN
  FOR index:= 1 to global_input_ids[] size
    global_input[index]:= (global_input_ids[index],
                                global_input_quantities[index]);
  END FOR

  FOR index:= 1 to global_output_ids[] size
    global_output[index]:= (global_output_ids[index],
                                global_output_quantities[index]);
  END FOR

  FOR index:= min_resource_id to max_resource_id
    available_resources[index]:= index;
  END FOR

  FOR each id in global_input_ids[]
    remove id from available_resources[];
  END FOR

  FOR each id in global_output_ids[]
    remove id from available_resources[];
  END FOR
END
```

The next two sections present in detail the steps of the subroutines that constitute the algorithm of the instance generator. For their implementation are used the following functions:

1. `random_ints(num, from, to)`: returns a stream of random integers of size *num* within the interval [*from, to*].
2. `random_doubles(num, from, to)`: returns a stream of random doubles of size *num* within the interval [*from, to*].
3. `random()`: returns a random double within the interval [*0, 1*).
4. `random(n)`: returns a random integer within the interval [*1, n*].
5. `round(d)`: rounds the double number *d* to the closer integer.
6. `distribute_full(resources[], tasks[])`: distributes all the resources in *resources[ ]* to tasks in *tasks[ ]*. If the resources are less than tasks then it duplicate some of the resources, if the resources are more than tasks then gives to some tasks more than one resource.
7. `distribute_part(resources[], tasks[])`: gives one of the resources in *resources[ ]* per task in *tasks[ ]*. If the resources are less than tasks then it duplicate some of the resources, if the resources are more than tasks then some resources remain into *available_resources[ ]*.

5.2.1 Biased task set creation

The first step is to create the body of the biased tasks and the values for the randomly calculated attributes, namely id, cost, revenue, duration, quality, flexibility and mutual exclusions. The created tasks are inserted to a temporary array *temp_biased_tasks[process_size]*.

```
IF process_size > 0 DO
  FOR index:= min_task_id to max_task_id
    task_ids[index]:= index;
  END FOR

  cost[]:= random_doubles(process_size, min_cost, max_cost);

  FOR index:= 1 to cost[] size
    revenue[index]:= cost[index] + cost[index] *0.2 + random() *
    revenue_margin;
  END FOR

  duration[] = random_ints(process_size, min_duration, max_duration);
  quality[] = random_ints(process_size, 1, quality_scale);
  flexibility[] = random_ints(process_size, 1, flexibility_scale);

  FOR index:= 1 to process_size
    create new biased_task(task_ids[index], cost[index],
    revenue[index], duration[index], quality[index],
    flexibility[index]);

    IF exists mutual_exclusions_list for task_ids[index]
```

```
      add mutual_exclusions_list to task;
    End IF

    temp_biased_tasks[index]:= biased_task;
  END FOR
```

At the beginning the set of the available resources is the initial input. The algorithm selects [1 to available resources number] tasks, distributes the available resources as their input and creates the necessary arcs for the process graph.

```
  current_input[] = global_input[];
  indices[] = random_ints (global_input[] size, 1, global_input[]
size);

  FOR each index in indices[]
    selected_tasks[]:= temp_biased_tasks[index];
  END FOR

  FOR index:= 1 to selected_tasks[] size
    links[index]:= (-1, selected_tasks[index].task.id);
  END FOR

  BEGIN BLOCK_1
    IF inter_connection == false:
      distribute_full(current_input[], selected tasks[]);
    Else
      distribute_part(current_input[], selected tasks[]);
    End If;

    remove consumed resources from current_input[];
  END BLOCK_1
```

Next in order to ensure connectivity, selects one of the aforementioned tasks, creates resources as its output, inserts this output to available resources, removes the selected task from the temporary array, inserts it to the permanent set of biased tasks and reduces the process_size by one.

```
  BEGIN BLOCK_2
    index:= random(selected_tasks[] size);
    select_task:= temp_biased_tasks[index];

    indices[]:= random_ints(resource_factor, 1, available_resources[]);

    FOR each index in indices[]
      resource_id:= available_resources[index];
      resource_quantity:= random_doubles(1, min_resource_quantity,
                                        max_resource_quantity);
      task_output[index]:= (resource_id, resource_quantity);
    END FOR

    add task_output[] to select_task;
    add select_task into biased_tasks[];
    current_process_size:= process_size - 1;
  END BLOCK_2
```

While there are more than [final resources number] tasks, the algorithm sequentially selects randomly tasks according to the procedure that described in 4.1 using the formula (1) and connects them with the last inserted to the set of biased tasks in terms of output → input resources, creating the necessary arcs for the process graph. Then, selects randomly one of them to be inserted to the set of biased tasks and repeats the step sequence.

```
  WHILE temp_biased_tasks[] size > global_output[] size:
    task_connections =  round(current_process_size * net_complexity);
    biased_task:= biased_tasks[biased_tasks size];

    FOR each out_resource in biased_task.task_output[]
      add out_resource into current_input[];
    END FOR

    indices[]:= random_ints(task_connections, 1,temp_biased_tasks[]
    size);

    FOR each index in indices[]
      selected_tasks[index]:= temp_biased_tasks[index];
    END FOR

    FOR each task in selected_tasks[]
      add into links[] (biased_task.id, selected_tasks[].task.id);
    END FOR

    REPEAT BLOCK_1;
    REPEAT BLOCK_2;
  END WHILE
```

Finally, the currently available resources are distributed to the last [final resources number] tasks as input and the final resources as output, creating at the same time the last arcs for the process graph.

```
  biased_task:= biased_tasks[biased_tasks size];

  FOR each out_resource in biased_task.task_output[]
      add out_resource into current_input[];
  END FOR

  FOR index:= 1 to temp_biased_tasks[] size
    selected_tasks[index]:= temp_biased_tasks[index];
  END FOR

  FOR each task in selected_tasks[]
    add into links[] (biased_task.id, selected_tasks[].task.id);
    add into links[] (selected_tasks[].task.id, -2);
  END FOR

  REPEAT BLOCK_1;
  distribute_full(global_output[], selected tasks[]);

  FOR each task in selected_tasks[]
      add task into biased_tasks[];
  END FOR
END IF
```

5.2.2 Arbitrary task set generator

The creation of the set of arbitrary tasks uses the same sequence of steps, except that everything is calculated randomly, the evaluation of the attributes is deliberately worst and the initial input/final output resources are excludes from the set of the available resources, to ensure that the arbitrary tasks cannot participate to the construction of an optimum business process design.

```
 IF population > 0
  FOR index:= max_task_id +1 to population
    task_ids[index]:= index;
  END FOR

  cost[]:= random_doubles(population, (min_cost + min_cost * 0.2),
                                      (max_cost + max_cost * 0.2));

  FOR index:= 1 to cost[] size
    revenue[index]:= cost[index] - cost[index] *0.2 - random() *
revenue_margin;
  END FOR

  duration[] = random_ints(population, (min_duration +
           min_duration * 0.2), (max_duration + max_duration * 0.2));
  quality[] = random_ints(population, 0, (quality_scale - 2));
  flexibility[] = random_ints(population, 0, (flexibility_scale - 2));

  FOR index:= 1 to population
    create new arbitrary_task(task_ids[index], cost[index],
revenue[index],
                    duration[index], quality[index],
flexibility[index]);
    arbitrary_tasks[index]:= arbitrary_task;
  END FOR

  FOR index:=1 to population
    input_num:= random(2);
    indices[]:= random_ints(input_num, 1, available_resources[]);

    FOR each index in indices[]
      resource_id:= available_resources[index];
      resource_quantity:= random_doubles(1, min_resource_quantity,
                                            max_resource_quantity);
      task_input[index]:= (resource_id, resource_quantity);
    END FOR

    add task_input[] to arbitrary_task[index];
    output_num = random(2);
    indices[]:= random_ints(output_num, 1, available_resources[]);

    FOR each index in indices[]
      resource_id:= available_resources[index];
      resource_quantity:= random_doubles(1, min_resource_quantity,
                                           max_resource_quantity);
      task_output[index]:= (resource_id, resource_quantity);
    END FOR

    add task_output[] to arbitrary_task[index];
  END FOR
END IF
```

5.2.3 Auxiliary functionalities

The Task generator application offers three optional functionalities to facilitate the better understanding of the constructed feasible business process and the usability with the processes optimization application. Specifically, the visualization of the graph of the feasible business process network can act as clear overview of the construction logic. Additionally, there is the possibility to print the calculated result into a text file for every intended use. Finally, by default the input resources of the tasks are sorted in ascending order according their id the application and the user can choose out of two types of task sorting during printing into a file. That is, the constructed tasks can be printed sorted in ascending order in the base of their id value or in ascending order in the base of the id of their first input resource. The later is a characteristic of crucial importance for the process optimization algorithm.

5.3 Process network construction

The structure of the business process design and the precedence relations between the tasks can be represented by an acyclic, connected and directed graph. Specifically, the tasks, their input/output resources and their attributes are represented by the nodes and the precedence relations are represented by the directed arcs that connect the nodes. The principle for the construction of the process graph is that for each node v there is a directed path from the dummy start-node towards v and a directed path from v towards the dummy end-node, therefore each node, except the start-node, has at least one predecessor and except the end-node, at least one successor (Kolisch, Sprecher and Drexl, 1995; Kolisch and Padman, 2001). In more detail, the constructive procedure that ensures the connectivity of the process design graph is conducted as follows:

1. When a task T_i consumes as input a resource that is output of the dummy start task, then is added to the graph an arc from the start-node towards the node that represents T_i with the label (Start → T_i_id).
2. When task T_i produces as output a resource that is input for the task T_j, then is added to the graph an arc from the node that represents T_i toward the node that represents T_j, with the label (T_i_id → T_j_id).
3. When a task T_i produces as output a resource that is input for the dummy end task, then is added to the graph an arc from the node that represents T_i toward the end-node with the label (T_i_id → End).

5.4 Application architecture

The presentation of the software architecture is completed with the necessary UML diagrams of designing level. The following *Use Case Diagram* (Figure 5.5) demonstrates graphically the structural architecture, the dependencies and the interactions between the main consisting elements of the application.

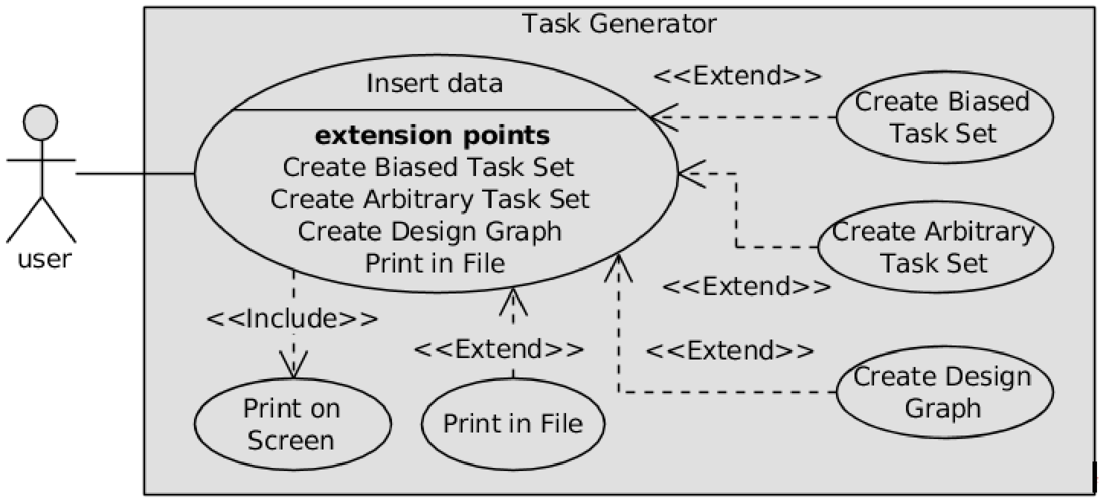


Figure 5.5.Use Case diagram

In addition, the *Activity Diagram* (Figure 5.6) shows the execution sequence of the generator and the possible paths of the control flow.

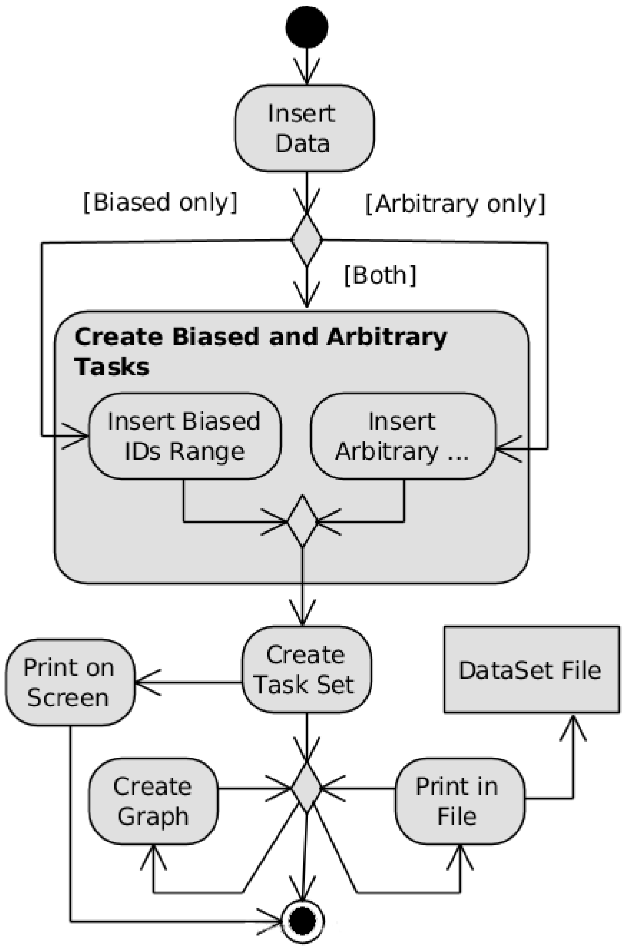


Figure 5.6.Activity diagram

The *Class Diagram* (Figure 5.7) depicts a static view of the application, presenting the main classes and the associations between them.

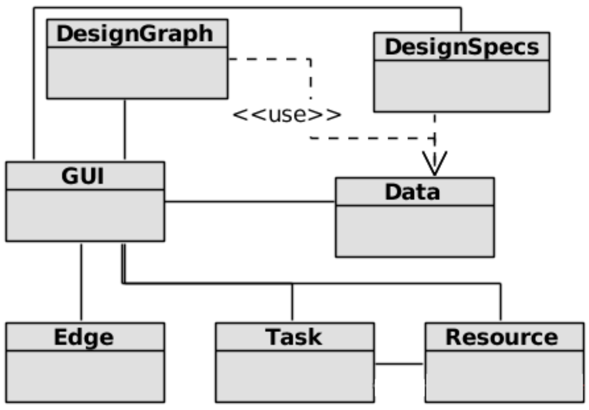


Figure 5.7.Class diagram

Finally, the *Sequence Diagram* (Figure 5.8) exhibits the time dimension by describing chronically the sequence of the steps and decisions for a complete functional circle of the application.

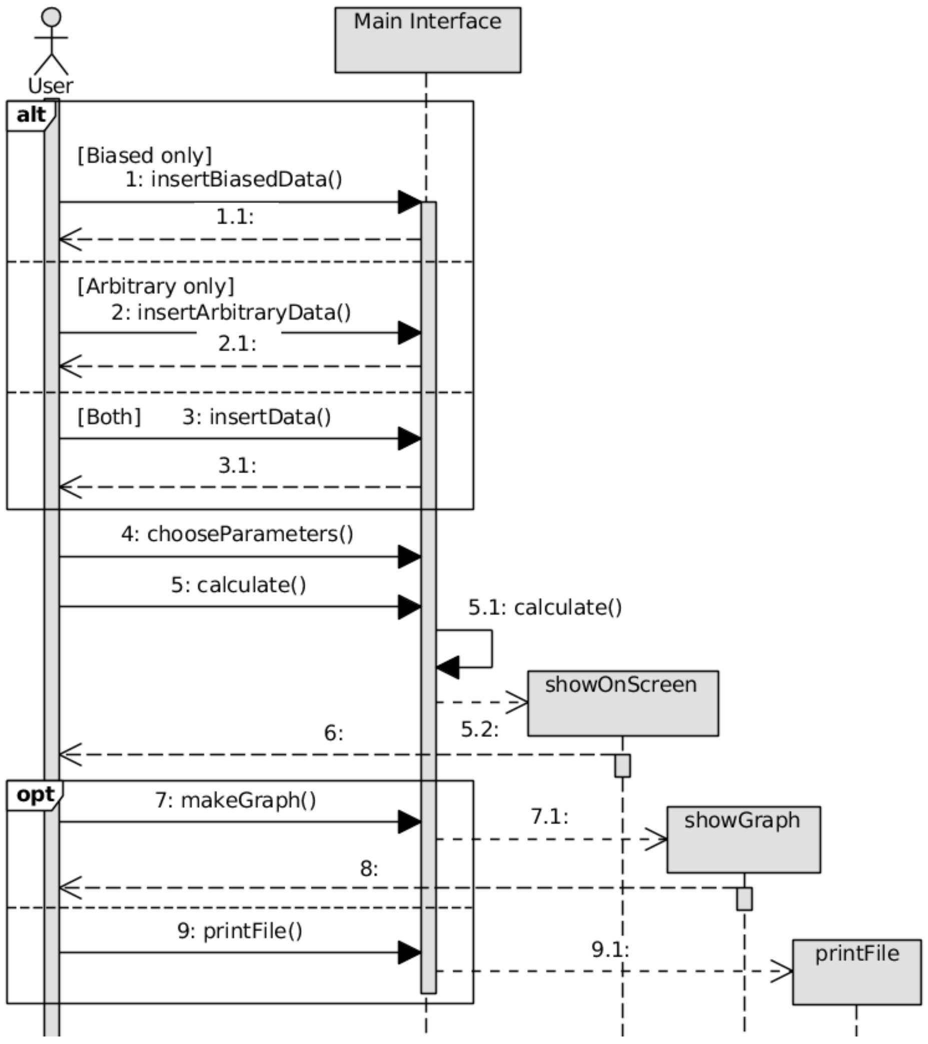
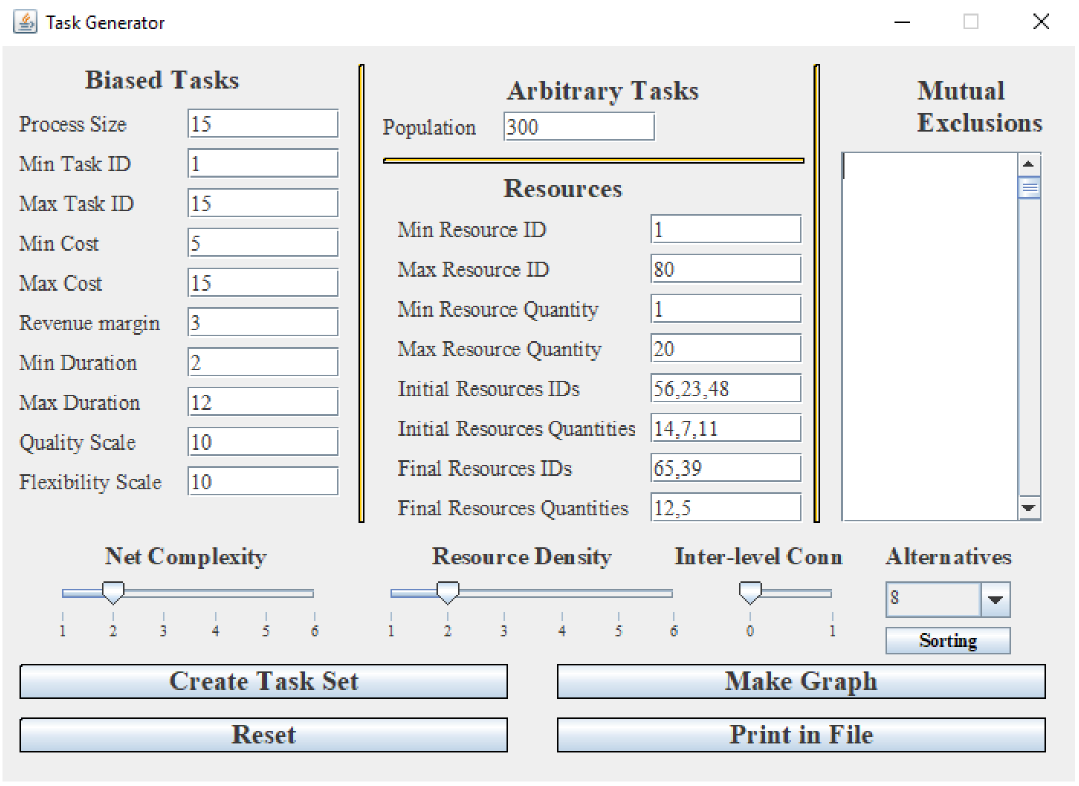


Figure 5.8.Sequence diagram

5.5 Experimental evaluation

For the more comprehensive understanding of the instance generator functionality, we present a step-by-step experimental execution with arbitrary input data and we display the produced outcomes. The first step for the user is to insert the necessary data and settings in the main interface of the application (Figure 5.9).



Task Generator

Biased Tasks

Process Size	15
Min Task ID	1
Max Task ID	15
Min Cost	5
Max Cost	15
Revenue margin	3
Min Duration	2
Max Duration	12
Quality Scale	10
Flexibility Scale	10

Arbitrary Tasks

Population: 300

Resources

Min Resource ID	1
Max Resource ID	80
Min Resource Quantity	1
Max Resource Quantity	20
Initial Resources IDs	56,23,48
Initial Resources Quantities	14,7,11
Final Resources IDs	65,39
Final Resources Quantities	12,5

Mutual Exclusions

Net Complexity: 2

Resource Density: 2

Inter-level Conn: 0

Alternatives: 8

Figure 5.9: Task Generator User Interface

In this experimental execution we want to create a feasible business process design of 15 biased tasks, 3 duplicated biased for the creation of $2^3=8$ alternative designs and 300 arbitrary tasks as supplement to the problem instance. The ids of the biased tasks are from 1 to 15 units, thus the ids of duplicated are from 16 to 18 and the ids of arbitrary tasks are from 19 to 319. The cost is calculated within the range [5, 15], while the margin for the revenue is 3. The execution time ranges from 2 to 12 units and for the evaluation of the attributes quality and flexibility is used a twenty-level scale from 1 to 10. Eighty different resource types can be utilized for the input/output of the tasks, with their quantity to range from 1 to 20 units.

The set of the initial input is consisted by the resources 56, 23, and 48 in quantities 14, 7 and 11 respectively, while the requested output set is consisted of the resources 65 and 39 in quantities 12 and 5 respectively. The Net Complexity factor was set to 2, meaning that each biased task can be related with 11% to 20% of the not yet processed biased tasks. The Resource Density factor was set to 2, thus each biased task outputs 1 to 3 different resources. Finally, the Inter-level Connection was set to 0 (false), therefore the immediate successors of a task belong to the same level of the process tree-graph. Table 5.4 presents the set of the generated biased task and in Figure 5.10 is depicted the graph of the business process design. The entire set of the generated instance can be found in Appendix C.

Table 5.4.Biased and Alternative tasks

Biased tasks
task(id(1),cost(10),revenue(18),duration(2),quality(8),flexibility(9),in([[1,4],[57,1]]),out([[44,7],[49,5]]),mutual_exclusions([])).
task(id(2),cost(7),revenue(14),duration(6),quality(1),flexibility(8),in([[12,2],[44,7],[49,5]]),out([[71,1]]),mutual_exclusions([])).
task(id(3),cost(12),revenue(20),duration(6),quality(9),flexibility(1),in([[12,2]]),out([[29,1],[34,5],[52,11]]),mutual_exclusions([])).
task(id(4),cost(14),revenue(20),duration(9),quality(3),flexibility(6),in([[31,3],[56,14],[58,1],[76,16]]),out([[17,4],[41,18]]),mutual_exclusions([])).
task(id(5),cost(10),revenue(12),duration(2),quality(8),flexibility(6),in([[59,12]]),out([[3,11],[66,5],[72,17]]),mutual_exclusions([])).
task(id(6),cost(11),revenue(19),duration(10),quality(7),flexibility(4),in([[21,6]]),out([[39,5]]),mutual_exclusions([])).
task(id(7),cost(14),revenue(20),duration(6),quality(5),flexibility(6),in([[29,1],[34,5]]),out([[68,14],[75,7],[79,6]]),mutual_exclusions([])).
task(id(8),cost(5),revenue(9),duration(11),quality(8),flexibility(8),in([[52,11],[68,14],[73,18],[79,6]]),out([[57,1]]),mutual_exclusions([])).
task(id(9),cost(12),revenue(14),duration(7),quality(7),flexibility(8),in([[17,4],[41,18]]),out([[21,13]]),mutual_exclusions([])).
task(id(10),cost(9),revenue(17),duration(4),quality(3),flexibility(7),in([[3,11],[66,5],[72,17]]),out([[73,18]]),mutual_exclusions([])).
task(id(11),cost(9),revenue(11),duration(10),quality(9),flexibility(2),in([[25,10],[48,11]]),out([[31,3],[58,1],[76,16]]),mutual_exclusions([])).
task(id(12),cost(5),revenue(9),duration(4),quality(10),flexibility(9),in([[71,1]]),out([[25,10]]),mutual_exclusions([])).
task(id(13),cost(10),revenue(15),duration(6),quality(8),flexibility(7),in([[75,7]]),out([[1,4],[59,12]]),mutual_exclusions([])).
task(id(14),cost(14),revenue(23),duration(7),quality(6),flexibility(10),in([[23,7]]),out([[12,7]]),mutual_exclusions([])).
task(id(15),cost(12),revenue(17),duration(11),quality(1),flexibility(8),in([[12,3],[21,7]]),out([[65,12]]),mutual_exclusions([])).
Alternative tasks
task(id(16),cost(8),revenue(16),duration(11),quality(8),flexibility(6),in([[25,10],[48,11]]),out([[31,3],[58,1],[76,16]]),mutual_exclusions([])).

task(id(17),cost(8),revenue(16),duration(6),quality(10),flexibility(6),in([[31,3],[56,14],[58,1],[76,16]]),out([[17,4],[41,18]]),mutual_exclusions([])).
task(id(18),cost(7),revenue(8),duration(6),quality(6),flexibility(6),in([[71,1]]),out([[25,10]]),mutual_exclusions([])).

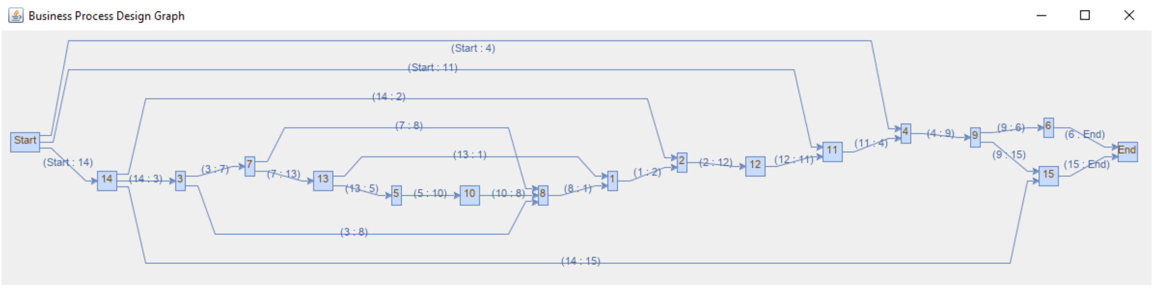


Figure 5.10.Business Process Design Graph

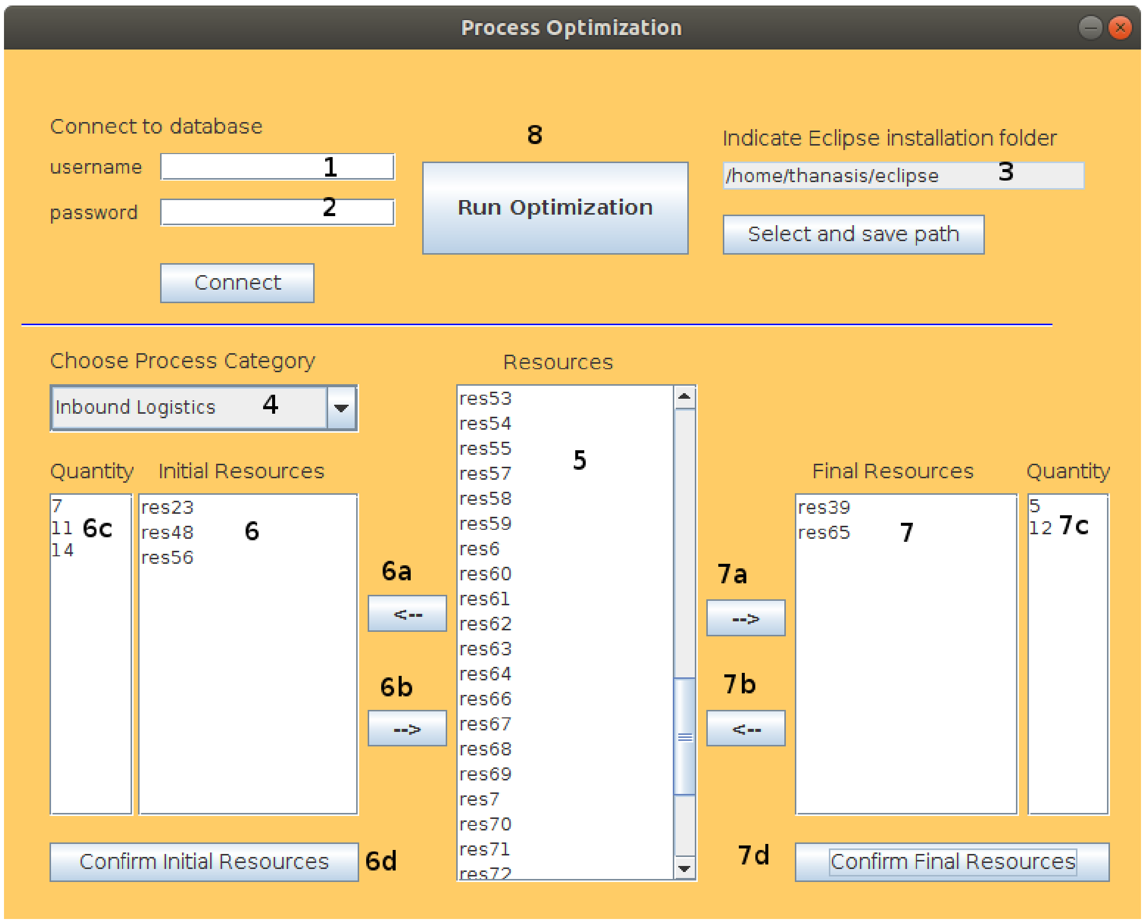
6 Experimental evaluation

The aim of the sixth chapter is twofold. In the first section, it presents a complete step-by-step execution of the developed software utilizing the necessary screenshots for a more comprehensive understanding. The second section tries to investigate the capabilities and limitations of the proposed algorithm, presenting an experimental analysis of the performance as function of the size of the problem's instance.

6.1 Execution example

For the demonstration of the functionality of the framework the problem's instance that was created by the *Task_Generator* is used, as described in the previous chapter. The produced tasks are stored into the database as members of the functional category *Inbound Logistics* and constitute the *Task Library.* Here we should remind the reader that the aforementioned instance consists of 318 tasks and contains eight feasible business process designs which each containing 15 tasks. Additionally, the *Resource Library* contains eighty resources from which the decision maker can select the initial input and the required output. The initial required step is the connection to the database using the credentials of a valid user (Figure 6.1, 1 and 2). Next follows the selection of the ECL^{i}PS^{e} installation directory (Figure 6.1, 3). For the specific example the algorithm's parameters was configured as follows:

1. Inbound Logistics as the category of the business process design (Figure 6.1, 3).
2. The initial input resources are the res23, res48, and res56 (Figure 6.1, 6) in quantities 7, 11 and 14 units respectively (Figure 6.1, 6c), selected from the available resources (Figure 6.1, 5) using the arrow-controllers (Figure 6.1, 6a and 6b) and confirming with the button *Confirm Initial Resources* (Figure 6.1, 6d).
3. The resources res39 and res65 (Figure 6.1, 7) in quantities 5 and 12 units respectively (Figure 6.1, 7c) as initially required output, selected from the available resources (Figure 6.1, 5) using the arrow-controllers (Figure 6.1, 7a and 7b) and confirming with the button *Confirm Final Resources* (Figure 6.1, 7d).



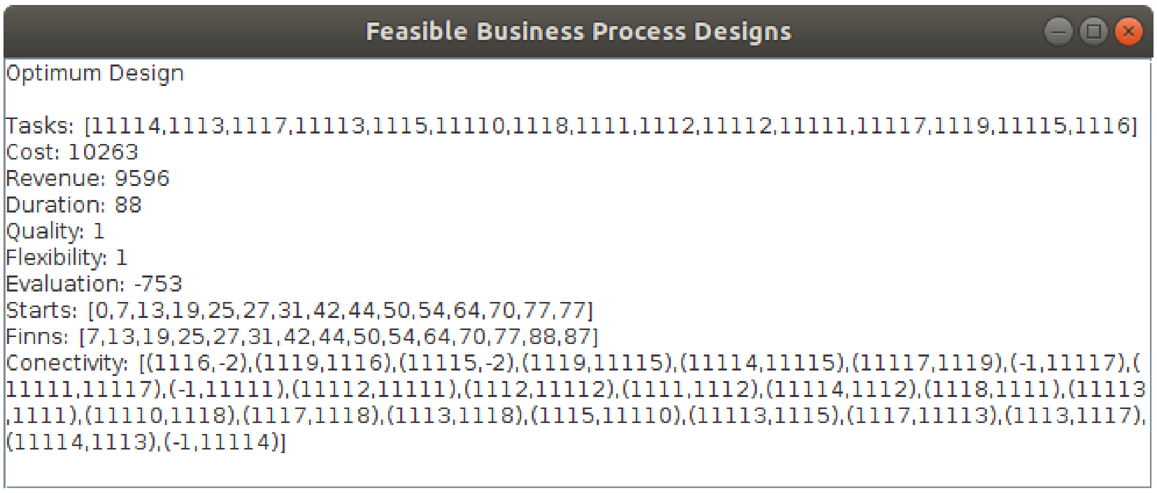
Quantity	Initial Resources
7	res23
11	res48
14	res56

Resources: res53, res54, res55, res57, res58, res59, res6, res60, res61, res62, res63, res64, res66, res67, res68, res69, res7, res70, res71, res72

Final Resources	Quantity
res39	5
res65	12

Figure 6.1.Initial screen

The last interaction by the user is the activation of button *Run Optimization* (Figure 6.1, 8), which initiates in the background the execution chain keeping it transparent from the decision maker. At the end of the computation, if the ECL^{i}PS^e infers a solution then the application retrieves the necessary data for the optimized design, prints on screen the information and depicts in a separate window the graph of the process design. For our example, the attributes of the constructed optimum design are shown in Figure 6.2, while the constructed graph is depicted in Figure 6.3.



Feasible Business Process Designs

```
Optimum Design

Tasks: [11114,1113,1117,11113,1115,11110,1118,1111,1112,11112,11111,11117,1119,11115,1116]
Cost: 10263
Revenue: 9596
Duration: 88
Quality: 1
Flexibility: 1
Evaluation: -753
Starts: [0,7,13,19,25,27,31,42,44,50,54,64,70,77,77]
Finns: [7,13,19,25,27,31,42,44,50,54,64,70,77,88,87]
Conectivity: [(1116,-2),(1119,1116),(11115,-2),(1119,11115),(11114,11115),(11117,1119),(-1,11117),(
11111,11117),(-1,11111),(11112,11111),(1112,11112),(1111,1112),(11114,1112),(1118,1111),(11113
,1111),(11110,1118),(1117,1118),(1113,1118),(1115,11110),(11113,1115),(1117,11113),(1113,1117),
(11114,1113),(-1,11114)]
```

Figure 6.2.Optimum design attributes

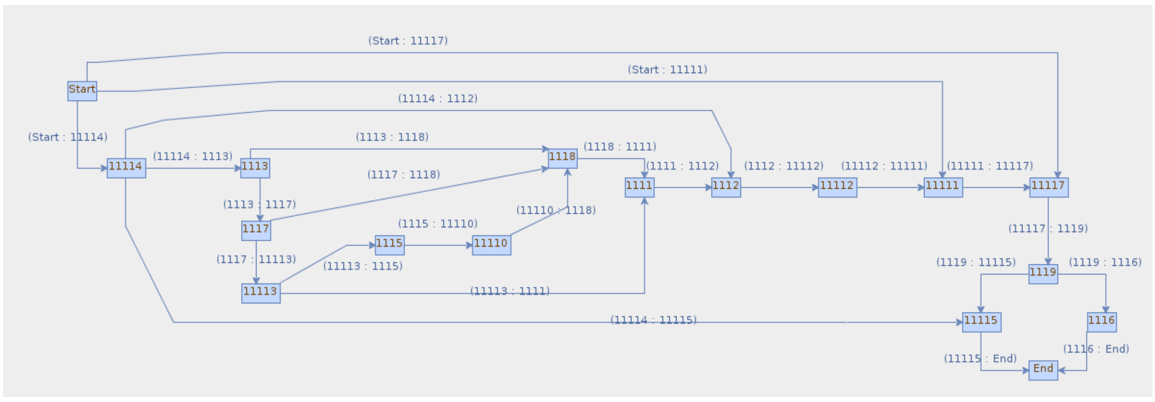


Figure 6.3.Optimum design graph

6.2 Performance evaluation

The target of the proposed optimization algorithm is twofold, initially to discover the tasks that can construct an optimum process design regarding the criteria cost, revenue, quality, flexibility and then to schedule the process in a way that minimizes the total execution time. This means that the algorithm tries to solve concurrently two NP-hard mathematical problems, namely the search and the scheduling problem. For this reason, we conducted two separate comparative evaluations, one to test the performance of the algorithm only in scheduling and the other to test the performance when both searching and scheduling are performed. Before we proceed, we provide a discussion of the meaning and the defining factors of the term *connectivity complexity*. More specifically, the aforementioned term refers to (1) the number of connections that a task has with other tasks in the context of a constructed process design, (2) the number of output resources of the tasks and (3) to the existence or not of task groups that are executed in parallel. Referring to *Task Generator*, the first characteristic is controlled by the *Network Complexity Factor* (NC) (Table 5.2), the second is defined by the *Resource Density Factor* (RD) (Table 5.3) and the third is determined by *Inter-level Connection Factor* (IC) (Figures 5.1, 5.2, 5.3 and 5.4). The higher are their values, the higher is the connectivity complexity.

6.2.1 Scheduling evaluation

For the evaluation of the algorithm regarding only scheduling, 24 instances of various process sizes and factor combinations were constructed using the *Task Generator* as shown in Table 6.1. The column *Design size* contains the number of tasks that must be scheduled, the columns *NC*, *RD*, *IC* shows the combinations of the connectivity complexity factors and the column *cpu time* the performance of the algorithm for the corresponding instance. The notation *+60* means that a sub-optimal solution was found, because the time expiration limit of the branch-and-bound algorithm.

Table 6.1.Scheduling evaluation

Scheduling evaluation										
Design size	**NC**	**RD**	**IC**	**cpu time**		**Design size**	**NC**	**RD**	**IC**	**cpu time**
30	1	2	0	0.03		90	1	2	0	0.17
30	1	2	1	+60		90	1	2	1	+60
30	3	5	0	0.03		90	3	5	0	0.73
30	3	5	1	+60		90	3	5	1	+60
30	4	6	0	0.05		90	4	6	0	1.05
30	4	6	1	+60		90	4	6	1	+60
60	1	2	0	0.02		120	1	2	0	0.36
60	1	2	1	+60		120	1	2	1	+60
60	3	5	0	0.19		120	3	5	0	1.80
60	3	5	1	+60		120	3	5	1	+60
60	4	6	0	0.28		120	4	6	0	2.94
60	4	6	1	+60		120	4	6	1	+60

The results show that the performance of proposed algorithm is not affected by the size of the process and can easily schedule large problems of 120 tasks when the connectivity complexity is medium to high. But when there are many groups of many parallely executed tasks and the connectivity complexity becomes very high (IC = 1), then the performance is very poor even for small process sizes of 30 tasks. Finally, regarding the *NC* and *RD* factors, the results show a rather small impact to the overall performance of the scheduling.

6.2.2 Searching and Scheduling evaluation

Similarly, for the evaluation of the algorithm regarding both searching and scheduling, 21 instances of various process sizes, total sizes and factor combinations were constructed using the *Task Generator* as shown in Table 6.2. Additionally to the columns of the Table 6.1, ithe column *Instance size* is used, that contains the total number of the task set and the column *Alt* that denotes the number of the alternative feasible designs. The absence of value in the *cpu time* column denotes that the algorithm failed to find any solutions within the time limit of 60 seconds.

Table 6.2.Search and Scheduling evaluation

Search and Scheduling evaluation														
Design size	**Instance size**	**Alt**	**NC**	**RD**	**IC**	**cpu time**		**Design size**	**Instance size**	**Alt**	**NC**	**RD**	**IC**	**cpu time**
20	523	8	1	2	0	0.13		60	211	2	4	6	0	48.41
20		0	1	2	1	+60		80	580	0	1	2	0	0.91
20		8	3	5	0	2.86		80	580	0	3	5	0	23.95
20		0	3	5	1	-		80	350	0	4	6	0	5.30
20		8	4	6	0	1.18		100	600	0	1	2	0	1.11
20		0	4	6	1	-		100	600	0	3	5	0	15.75
40	543	8	1	2	0	24.64		100	235	0	4	6	0	6.38
40	353	4	3	5	0	38.45		120	620	0	1	2	0	3.63
40	541	2	4	6	0	1.65		120	300	0	3	5	0	52.66
60	563	8	1	2	0	2.84		120	220	0	4	6	0	29.78
60	462	4	3	5	0	35.55								

The results show that the main affecting characteristic for the performance is the existence of a high number of groups of many parallely executed tasks. We observe that even for only one design (*Alt = 0*) of 20 tasks, when *IC = 1* the algorithm fails to find a solution within the time limit. The same negative effect seems to cause the existence of alternative feasible designs in the task set. Indeed, from a design size of 40 tasks the impact starts to become significant while from size 80 tasks and above the algorithm fails to find solution regardless the total size. This is the reason for the absence of results with *IC = 1* and *Alt > 0* for design sizes bigger than 40 and 80 respectively. On the other hand, when the connectivity complexity is medium to high the performance is satisfactorily even for combinations of large design sizes, total sizes and high values of *NC* and *RD* factors.

6.3 Summary

The sixth chapter was devoted to the description of how the software can be used and to the exploration of its capabilities. Thus, initially a complete step-by-step cycle of execution via screenshots is presented and then is conducted a comparative evaluation of the performance of the proposed algorithm, using problem instances of various sizes and connectivity complexity that was constructed with the *Task Generator* application.

7 Conclusion

The modern approaches of Business Administration consider the business processes as the crucial element for the organizational structure of an effective enterprise, for the achievement of high competitiveness and profitability. Therefore, the administration of processes has become a major concern for the companies and researchers in an attempt to develop methodologies that can assist processes' continuous improvement. The review of the relevant bibliography revealed that the majority of these methodologies are based on graphical paradigms, evolutionary algorithms or data mining techniques while is observed a complete absence of methods that are based on CLP. However, CLP is by nature a searching technique in the domain of decision variables under the limitation of predefined constraints. Therefore, CLP can offer the computational environment for the solution of complex combinatorial problems such as business process optimization.

7.1 Results

The present thesis investigated the applicability of CLP to the field of process optimization, by the design and development of a framework that integrates heterogeneous technologies in order to compose optimized business processes by simple tasks in the base of quantitative and qualitative evaluation criteria. The developed framework is consisted by a relational database for the storage of the task related data, a Java interface which handles the interaction between the subsystems and the user, the ECL^{i}PS^e platform that performs the optimization procedure utilizing logic programming and finally, the JGraphT which visualizes the constructed optimum process design as a acyclic directed graph. To deal with the fact that the business process optimization is considered as a NP-hard problem and the performance of an algorithm depends on the size of a specific instance (Wang, Salhi and Fraga, 2004;Georgoulakos et al., 2017), the proposed framework involves three techniques of data reduction.

The first is implemented in the system's database, where the tasks are organized by an indexing schema according to Porter's Value Chain that groups the tasks into non-overlapping categories and decreases the searching domain of the problem. The second data reduction technique is the exclusion of the tasks with certain characteristics in the base of predefined rules. Finally, is utilized a clustering technique for the separation of the input task set into twenty groups using as criterion the id of their first input resource, in order to be reduced the searching space in each iteration of the optimization algorithm. For the evaluation of the performance and the detection of the limitations of the proposed algorithm, was developed the Task Generator application which can create instances of various sizes of the formulated problem, manipulating their characteristics through the factors *Network Complexity*, *Resource Density* and *Interconnection*.

7.2 Limitations

The benchmarking with instances of various sizes and parameter configuration, reviled that the algorithm is suitable for problem's instances of small to medium size and low connectivity complexity when is required searching and scheduling of an optimized business process design. On the other hand, it seems to be able to confront efficiently instances of large sizes but low to medium complexity when only scheduling is required. Although the computational complexity is exponential, the clustering technique that is used to split up the input task set into twenty groups, speeds up satisfactorily the execution and succeeds to give exact solution to realistic problem instances within practical time. However, in the general case the proposed algorithm as an exact method has limitation to escalation, thus can be considered as a solution for small to medium instance sizes and limited connectivity complexity.

7.3 Future research

In the context of the proposed framework the tasks are manipulated as "black boxes" without any knowledge about their internal structure or functionality. The only known characteristics of a task are the input/output resources and the evaluation for the criteria cost, revenue, duration, quality and flexibility. This means that a constructed optimized process can be treated also as a task of a higher level. Taking advantage of this feature, we can create a multilevel hierarchy of tasks, stored into a corresponding multilevel hierarchy of tables in the database that preserves the constructing relationships between the tasks of each level with the tasks of the directly lower level. Considering that a very complex constructing procedure is usually consisted by smaller sub-procedures for reasons of better administrative control, the aforementioned strategy theoretically can be applied on business process optimization problems of any size.

Concerning the current structure of the framework, some interventions can induct further improvement in terms of flexibility of decision-making, completeness of analysis and technical integration. In more detail, a systematic analysis of a company's structure can lead to a further refinement of its operations and define functional sub-sectors that contain smaller number of tasks and reduce the overall complexity. Regarding flexibility, each evaluation criterion can be refined to a set of sub-criteria that are more representative and comprehensible. For example, the criterion *Cost* can be refined to the sub-criteria *Row Materials Cost*, *Auxiliary Materials Cost*, *Labor Cost*, *Machinery Cost*, *Transportation Cost*, *Storage Cost* and offer better understanding to the decision maker in order to recognize problems and suggest possible modifications.

Additionally, the criteria may be weighted in an effort to simulate in a better way the reality of a specific case and incorporate into the analysis the particular perception of the decision maker. In terms of analysis completeness and robustness of the multicriteria analysis, could be employed additional evaluation criteria. Finally, in terms of technological integration the connection of the database of the proposed system with other transactional databases of a company can ensure the acquisition of timely data about the available resources and tasks. Additionally, the implementation of the framework as a web service from a local server or through the utilization of the Cloud and Internet services, could guarantee the integrity and validity of the analysis, scale economies as a single deployment point and convenience in use with mobile devices.

Bibliography

Ahmadikatouli, A. and Aboutalebi, M., 2011. New Evolutionary Approach to Business Process Model Optimization. In: *Proceedings of International MultiConference of Engineers and Computer Scientist Vol II* (IMECS). Hong Kong, China, 16-18 March, 2011.

Apt, K.R. and Wallace, M., 2007. *Constraint Logic Programming using ECL^{i}PS^e*. Cambridge, UK: Cambridge University Press.

Bhutta, K.S. and Huq, F., 1999. Benchmarking–best practices: an integrated approach. *Benchmarking: An International Journal*, 6(3), pp.254-268.

Coello, C.A.C., Lamont, G.B. and Van Veldhuizen, D.A., 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd ed. New York: Springer.

Conger, S., 2010. Six Sigma and Business Process Management. In: Vom Brocke, J., and Rosemann, M., eds., 2010. *Handbook on business process management 1: Introduction, Methods, and Information Systems*. 2nd ed. Heidelberg: Springer-Verlag. pp.127-146.

Davenport, T.H., 1993. *Process Innovation: Reengineering Work through Information Technology*. Boston: Harvard Business School Press.

Dumas, M., Mendling, J., La Rosa, M. and Reijers H.A., 2013. *Fundamentals of Business Process Management*. Heidelberg: Springer-Verlag.

Georgoulakos, K., Vergidis, K., Tsakalidis, G. and Samaras, N., 2017. Evolutionary Multi-Objective Optimization of business process designs with pre-processing. In: *Evolutionary Computation (CEC), 2017 IEEE Congress*. Donostia / San Sebastian, Spain: IEEE. pp. 897-904.

Gerke, K., Petruch, K. and Tamm, G., 2010. Optimization of Service Delivery through Continual Process Improvement: A Case Study. In: Abramowicz, W., Alt, R., Fahnrich, K.P., Franczyk, B. and Maciaszek, L.A., eds. 2010. *INFORMATIK 2010, Business Process and Service Science-Proceedings of ISSS and BPSC, Lecture Notes in Informatics (LNI), P-177*. Bonn: LNI. pp. 94-107.

Godfrey, A.B., 1999. Total Quality Management. In: Joseph M. Juran, J.M., Godfrey, A.B, Robert E. Hoogstoel R.E., and Edward G. Schilling E.G., eds. 1999. *Juran's Quality Handbook*. New York: McGraw-Hill. Ch.14.

Groger, C., Schwarz, H., and Mitschang, B., 2014. Prescriptive Analytics for Recommendation-Based Business Process Optimization. In: *Proceedings of the 17th International Conference on Business Information Systems (BIS) 2014*. Larnaca: Springer. pp.25-37.

Hallerbach, A., Bauer, T. and Reichert, M., 2009. Guaranteeing soundness of configurable process variants in Provop. In: *Proceedings of 11th IEEE Conference on Commerce and Enterprise Computing (CEC 2009)*. Vienna: IEEE Computer Society Press. pp.98-105.

Han, J., Kamber, M. and Pei, J., 2012. *Data Mining Concepts and Techniques*. 3rd ed. Waltham: Morgan Kaufmann-Elsevier.

Heravizadeh, M., Mendling, J. and Rosemann, M., 2009. Dimensions of Business Processes Quality (QoBP). In: Ardagna, D., Mecella, M. and Yang, J., eds. 2009. *BPM 2008 Workshops, LNBIP 17*. Heidelberg: Springer-Verlag. pp.80-91.

Hammer, M. and Champy, J., 1993. *Reengineering the Corporation: A Manifesto for Business Revolution*. London: Brealey.

Hammer, M., 2015. What is Business Process Management? In: Vom Brocke, J. and M. Rosemann, M., eds. 2015. *Handbook on Business Process Management 1*, International Handbooks on Information Systems. 2nd ed. Heidelberg: Springer-Verlag. pp.3-16.

Havey, M., 2005. *Essential Business Process Modeling.* O'Reilly Media Inc.

Hofacker, I. and Vetschera, R., 2001. Algorithmical approaches to business process design. *Computers & Operations Research*, 28, pp.1253-1275.

Kaplan, R.S. and Norton, D.P., 1992. The balanced scorecard: Measures that drive performance. *Harvard Business Review*, 70(1), pp.71-79.

Knuth, D. 1998. *Sorting and searching. The Art of Computer Programming Volume 3.* 2nd ed. Reading, Massachusetts: Addison-Wesley. ISBN 0-201-89685-0.

Kohonen, T., 1995. *Self-Organizing Maps*. Heidelberg: Springer-Verlag.

Kolisch, R. and Padman, R., 2001. An integrated survey of deterministic project scheduling. *Omega*, 29(3), pp. 249-272.

Kolisch, R., Schwindt, C., and Sprecher, A., 1999. Benchmark instances for project scheduling problems. In: J. Weglarz ed. 1999. *Project scheduling: Recent Models, Algorithms and Applications.* Boston MA: Springer. pp. 197-212.

Kolisch, R., Sprecher, A. and Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, 41(10), pp. 1693-1703.

Li, C., Reichert, M. and Wombacher, A., 2010. The MinAdept clustering approach for discovering reference process models out of process variants. *International Journal of Cooperative Information Systems*, 19(3-4), pp.159-203.

Lohrmann, M., and Reichert, M., 2013. Understanding business process quality. In: Glykas, M., Ed, 2013. *Business Process Management, Theory and Applications*. Heidelberg: Springer-Verlag. pp.41-73.

Melao, N. and Pidd, M., 2000. A conceptual framework for understanding business process modeling. *Information System Journal*, 10, pp.105-129.

Moon, C. and Seo, Y., 2005. Evolutionary algorithm for advanced process planning and scheduling in a multi-plant. *Computers & Industrial Engineering*, 48(2), pp.311-325.

Naveh, B. and Contributors, 2018. JGraphT. [online] Available at: <https://jgrapht.org> [Accessed 29 August 2018)]

Niederlinsky, A., 2014. *A Gentle Guide to Constraint Logic Programming via ECL^{i}PS^e*. 3rd ed. Gliwise: Jacek Skalmierski Computer Studio.

Niedermann, F. and Schwarz, H., 2011. Deep Business Optimization: Making Business Process Optimization Theory Work in Practice. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R. and Bider, I., eds. 2011. *BPMDS 2011 and EMMSAD 2011, LNBIP, 81*. Heidelberg: Springer-Verlag. pp. 88-102.

Oaklant J, 2003. *Total Quality Management: text with cases*. 3rd ed. Amsterdam: Butterworth-Heinemann.

Omachonu, V.K, and Ross, J.E., 2004. *Principles of total quality*. 3rd ed. New York: CRC Press LLC.

O'Neill, P., and Sohal, A.S., 1999. Business Process Reengineering. A review of recent literature. *Technovation*, 19(9), pp.571-581.

Porter, M. E., 1985. Competitive advantage: creating and sustaining superior performance. New York: Nova Science Publishers.

Reichert M., Rinderle-Ma S. and Dadam P., 2009. Flexibility in Process-Aware Information Systems. In: Jensen K. and Van der Aalst W.M.P., eds. 2009. *Transactions on Petri Nets and Other Models of Concurrency II. Lecture Notes in Computer Science*, 5460. Heidelberg: Springer-Verlag. pp.115-135.

Rossi, F., Van Beek, P. and Walsh, T., eds. 2006. *Handbook of Constraint Programming*. Amsterdam: Elsevier.

Tiwari, A., Vergidis, K. and Majeed, B., 2006a. Evolutionary Multi-objective Optimisation of Business Processes. In: *Proceedings of IEEE Congress on Evolutionary Computation 2006*. Vancouver: IEEE. pp.3091-3097.

Tiwari, A., Vergidis, K. and Turner, C., 2010. Evolutionary Multi-Objective Optimization of Business Processes. *Soft Computing in Industrial Applications*, pp. 293-301.

Valiris, G. and Glykas, M., 2004. Business analysis metrics for business process redesign. *Business Process Management Journal*, 10(4), pp. 445-480.

Van der Aalst, W. M. P. and Van Hee, K. M., 1996. Business process redesign: A Petri-net-based approach. *Computers in industry*, 29(1), pp. 15-26.

Van der Aalst, W.M.P., 1998. The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1), pp. 21-66.

Van der Aalst, W. M. P., 2004. Business process management demystified: a tutorial on models, systems and standards for workflow management. In: Desel, J., Reisig, W. and Rozenberg, G., eds. 2004. *Lectures on Concurrency and Petri Nets*, vol. 3098 of *Lecture Notes in Computer Science*. Heidelberg: Springer-Verlag. pp.1-65.

Van der Aalst, W.M.P., 2013. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, Vol 2013, Article ID 507984, 37 pages. Hindawi Publishing Corporation.

Van der Aalst, W. M. P., ter Hofstede A. H. M. and Weske, M., 2003. Business process management: a survey. In: van der Aalst, W. M. P., ter Hofstede, A. H. M., and Weske, M., eds. 2003. *Proceedings of the International Conference on Business Process Management (BPM '03)*, Vol. 2678 of *Lecture Notes in Computer Science (LNCS)*. Heidelberg: Springer-Verlag. pp.1-12.

Vergidis, K., Tiwari, A. and Majeed, B., 2007. Composite Business Processes: An Evolutionary Multi-objective Optimisation Approach. In: *Evolutionary Computation, 2007, CEC 2007, IEEE Congress*. Singapore: IEEE. pp.2672-2678.

Vergidis, K., Tiwari, A. and Majeed, B., 2006b. Business Process Improvement using Multi-objective Optimisation. *BT Technology Journal*, 24(2), pp.229-235.

Vergidis, K. and Tiwary, A., 2008. Business process design and attribute optimization within an evolutionary framework. In: *Proceeding of 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. Hong Kong, China, 1-6 June: IEEE.

Volkner, P. and Werners, B., 2000. A decision support system for business process planning. *European Journal of Operational Research*, 125, pp. 633-647.

Vom Brocke, J., and Rosemann, M., eds. 2010. *Handbook on business process management 1: Introduction, Methods, and Information Systems.* 2nd ed. Heidelberg: Springer-Verlag.

Wang, K., Salhi, A. and Fraga, E.S., 2004. Process design optimization using embedded hybrid visualization and data analysis techniques within a genetic algorithm optimization framework. *Chemical Engineering and Processing*, 43, pp.663-675.

Weske, M., 2012. *Business Process Management: Concepts, Languages, Architectures*. Heidelberg: Springer-Verlag.

Wibig, M., 2013. Dynamic Programming and Genetic Algorithm for Business Processes Optimisation. *I.J. Intelligent Systems and Applications,* 01, pp.44-51.

Wikipedia contributors, 2018. Factorial. In: *Wikipedia, The Free Encyclopedia*. [online] Available at: <https://en.wikipedia.org/w/index.php?title=Factorial&oldid=853729291> [Accessed 28 August 2018].

Wikipedia contributors, 2019. Binary search algorithm. In: *Wikipedia, The Free Encyclopedia*. [online] Available at: <https://en.wikipedia.org/w/index.php?title=Binary_search_algorithm&oldid=880095920> [Accessed 30 January 2019].

Womack, J.P. and Jones, D.T., 2003. *Lean Thinking*. New York: Free Press.

Appendix A - Database

The thesis have as attachment the file *process_optimization.sql* that contains the SQL code for the creation of database that is used as component of the proposed framework for the implementation of its functionality. The code is provided “AS IS’ without any warranty by the writer of this thesis. Possible use involves the assumption of full responsibility by the user. The database developed and tested with the DBMS MySQL 8.0.

Appendix B - Java GUI

The thesis have as attachments the folders *Process Optimization* and *Task Generator* that contain the executable java files *Process_Optimization.jar* and *TaskGenerator.jar* respectively along with the necessary java libraries and the logic program that was constructed for the experimental evaluation of the application. The applications requires Java 8.0 or higher and is provided "AS IS' without any warranty by the writer of this thesis. Possible use involves the assumption of full responsibility by the user. The application developed and tested on Linux Ubuntu 18.04 LTS.

Appendix C - Arbitrary tasks

```
task(id(19),cost(12),revenue(10),duration(6),quality(5),flexibility(5),in([[2,4],[31,16],[54,2],[65,2],
[68,19]]),out([[24,13],[31,6]]),mutual_exclusions([])).
task(id(20),cost(9),revenue(7),duration(7),quality(7),flexibility(6),in([[32,5],[54,12],
[57,10]]),out([[33,2]]),mutual_exclusions([])).
task(id(21),cost(6),revenue(5),duration(13),quality(2),flexibility(2),in([[19,9],[30,3],[72,12]]),out([[33,13],
[52,2]]),mutual_exclusions([])).
task(id(22),cost(15),revenue(9),duration(13),quality(4),flexibility(1),in([[26,5],[42,17],[48,4],[56,3]]),out([[6,10],
[47,9]]),mutual_exclusions([])).
task(id(23),cost(12),revenue(10),duration(8),quality(1),flexibility(0),in([[2,16],[35,4],[53,15]]),out([[13,2],[24,18],
[61,14]]),mutual_exclusions([])).
task(id(24),cost(15),revenue(9),duration(4),quality(2),flexibility(0),in([[1,18],[11,2],[17,11],[67,3]]),out([[4,7],
[50,16]]),mutual_exclusions([])).
task(id(25),cost(14),revenue(8),duration(12),quality(2),flexibility(3),in([[38,11],[67,11]]),out([[25,3],
[70,19]]),mutual_exclusions([])).
task(id(26),cost(15),revenue(9),duration(4),quality(5),flexibility(5),in([[9,7],[42,11],[44,5],[58,13]]),out([[8,11],
[50,1]]),mutual_exclusions([])).
task(id(27),cost(11),revenue(9),duration(10),quality(0),flexibility(6),in([[2,7],[21,17],[27,13],
[45,8]]),out([[44,19]]),mutual_exclusions([])).
task(id(28),cost(6),revenue(5),duration(10),quality(6),flexibility(5),in([[35,11],[74,4]]),out([[13,18],[37,8],
[57,1]]),mutual_exclusions([])).
task(id(29),cost(8),revenue(6),duration(6),quality(7),flexibility(1),in([[15,1],[49,8],
[50,8]]),out([[7,9]]),mutual_exclusions([])).
task(id(30),cost(17),revenue(14),duration(9),quality(7),flexibility(5),in([[29,17],[34,9],[40,1],[56,13],
[61,7]]),out([[36,11],[66,12]]),mutual_exclusions([])).
task(id(31),cost(12),revenue(7),duration(9),quality(2),flexibility(6),in([[17,4],
[65,12]]),out([[26,13]]),mutual_exclusions([])).
task(id(32),cost(6),revenue(5),duration(10),quality(7),flexibility(7),in([[64,14],[73,6]]),out([[21,19],[37,13],
[42,3]]),mutual_exclusions([])).
task(id(33),cost(12),revenue(10),duration(5),quality(1),flexibility(6),in([[7,1],[35,19],[41,13]]),out([[39,9],
[62,6]]),mutual_exclusions([])).
task(id(34),cost(12),revenue(10),duration(9),quality(7),flexibility(5),in([[17,18],[69,9],[73,17]]),out([[26,6],[27,3],
[49,3]]),mutual_exclusions([])).
task(id(35),cost(13),revenue(7),duration(8),quality(4),flexibility(3),in([[29,4],[54,16]]),out([[14,10],[46,14],
[69,7]]),mutual_exclusions([])).
task(id(36),cost(17),revenue(11),duration(12),quality(3),flexibility(4),in([[25,5],[36,10],[52,18],[54,9],
[63,5]]),out([[7,8],[8,2],[65,11]]),mutual_exclusions([])).
task(id(37),cost(8),revenue(6),duration(6),quality(5),flexibility(4),in([[9,15],[34,4],[67,9]]),out([[34,4],
[54,13]]),mutual_exclusions([])).
task(id(38),cost(15),revenue(12),duration(8),quality(6),flexibility(6),in([[12,13],[28,11],[30,2],[36,7]]),out([[12,18],
[43,17],[52,7]]),mutual_exclusions([])).
task(id(39),cost(8),revenue(3),duration(12),quality(4),flexibility(3),in([[10,6],[25,16],[33,11],
[66,16]]),out([[22,16]]),mutual_exclusions([])).
```

```
task(id(40),cost(13),revenue(7),duration(6),quality(1),flexibility(6),in([[5,15],[23,18],[55,7],[64,3]]),out([[8,1],[21,14],
[69,6]]),mutual_exclusions([])).
task(id(41),cost(16),revenue(13),duration(10),quality(6),flexibility(1),in([[8,14],
[19,4]]),out([[63,18]]),mutual_exclusions([])).
task(id(42),cost(16),revenue(13),duration(8),quality(6),flexibility(5),in([[5,1],[13,17],[33,18],[51,6],
[72,18]]),out([[25,11],[71,10]]),mutual_exclusions([])).
task(id(43),cost(9),revenue(4),duration(7),quality(2),flexibility(2),in([[7,18],[29,2],[42,10],[67,1]]),out([[2,5],
[29,14]]),mutual_exclusions([])).
task(id(44),cost(6),revenue(5),duration(9),quality(1),flexibility(3),in([[11,17],[47,10]]),out([[51,7],
[71,19]]),mutual_exclusions([])).
task(id(45),cost(13),revenue(7),duration(7),quality(1),flexibility(6),in([[3,13],[6,2],[19,3],[65,16],
[68,18]]),out([[19,19],[21,6],[42,4]]),mutual_exclusions([])).
task(id(46),cost(9),revenue(4),duration(7),quality(6),flexibility(6),in([[5,17],[19,16],[28,19],[41,17],
[73,11]]),out([[10,12],[24,19]]),mutual_exclusions([])).
task(id(47),cost(10),revenue(8),duration(4),quality(5),flexibility(7),in([[44,10],
[74,7]]),out([[30,19]]),mutual_exclusions([])).
task(id(48),cost(14),revenue(11),duration(12),quality(6),flexibility(6),in([[36,15],
[62,6]]),out([[72,19]]),mutual_exclusions([])).
task(id(49),cost(11),revenue(9),duration(11),quality(0),flexibility(3),in([[17,5],[19,6],[42,1],[43,7]]),out([[14,19],
[70,14]]),mutual_exclusions([])).
task(id(50),cost(10),revenue(5),duration(4),quality(0),flexibility(7),in([[6,17],[8,5],[48,17],[61,6]]),out([[41,13],
[72,19]]),mutual_exclusions([])).
task(id(51),cost(12),revenue(10),duration(9),quality(3),flexibility(3),in([[6,9],[18,17],[27,11],
[42,11]]),out([[2,6]]),mutual_exclusions([])).
task(id(52),cost(17),revenue(14),duration(5),quality(7),flexibility(7),in([[18,12],[41,3],[65,11]]),out([[2,6],[47,8],
[74,17]]),mutual_exclusions([])).
task(id(53),cost(9),revenue(4),duration(7),quality(6),flexibility(3),in([[22,16],[35,8],[49,2],[55,18],
[74,12]]),out([[32,19]]),mutual_exclusions([])).
task(id(54),cost(17),revenue(14),duration(9),quality(0),flexibility(3),in([[22,10],[23,4],[30,2],[63,8],
[74,1]]),out([[31,5],[49,11]]),mutual_exclusions([])).
task(id(55),cost(11),revenue(9),duration(13),quality(3),flexibility(7),in([[11,9],[18,9],[22,3],[31,9],
[56,12]]),out([[12,13],[29,19],[62,7]]),mutual_exclusions([])).
task(id(56),cost(10),revenue(5),duration(10),quality(6),flexibility(3),in([[38,7],[41,5],[53,6],[59,1],
[67,15]]),out([[24,9],[30,9],[37,5]]),mutual_exclusions([])).
task(id(57),cost(11),revenue(9),duration(8),quality(5),flexibility(2),in([[26,16],[32,9],[34,11],[47,1],
[52,14]]),out([[39,1],[53,5]]),mutual_exclusions([])).
task(id(58),cost(11),revenue(6),duration(5),quality(2),flexibility(0),in([[25,14],[33,8],[34,2],[37,18],
[46,14]]),out([[40,12],[41,5],[73,9]]),mutual_exclusions([])).
task(id(59),cost(6),revenue(2),duration(6),quality(3),flexibility(6),in([[15,8],[16,15],[38,15],[45,8]]),out([[12,7],
[69,8]]),mutual_exclusions([])).
task(id(60),cost(15),revenue(12),duration(9),quality(1),flexibility(1),in([[39,17],[40,4],[46,19]]),out([[34,5],[46,3],
[52,16]]),mutual_exclusions([])).
task(id(61),cost(16),revenue(10),duration(8),quality(5),flexibility(1),in([[29,18],[32,8],[36,11],[62,3],
[68,6]]),out([[59,10],[61,4],[67,17]]),mutual_exclusions([])).
```

```
task(id(62),cost(15),revenue(9),duration(10),quality(1),flexibility(2),in([[52,10],[55,12]]),out([[25,14],
[38,2]]),mutual_exclusions([])).
task(id(63),cost(16),revenue(10),duration(5),quality(3),flexibility(2),in([[22,5],[32,12],[54,11],[58,6]]),out([[7,9],
[27,4],[60,1]]),mutual_exclusions([])).
task(id(64),cost(9),revenue(4),duration(8),quality(6),flexibility(4),in([[50,7],[63,13]]),out([[30,1],
[56,4]]),mutual_exclusions([])).
task(id(65),cost(14),revenue(11),duration(8),quality(4),flexibility(1),in([[33,12],[35,10],[57,11]]),out([[46,7],
[55,11]]),mutual_exclusions([])).
task(id(66),cost(16),revenue(13),duration(13),quality(3),flexibility(0),in([[11,17],[18,14],
[44,5]]),out([[55,6]]),mutual_exclusions([])).
task(id(67),cost(11),revenue(6),duration(9),quality(0),flexibility(1),in([[20,13],[51,3],[61,11]]),out([[9,18],[33,10],
[59,14]]),mutual_exclusions([])).
task(id(68),cost(7),revenue(3),duration(10),quality(6),flexibility(4),in([[29,15],[33,14],[37,5],[42,17],
[47,4]]),out([[67,11]]),mutual_exclusions([])).
task(id(69),cost(12),revenue(10),duration(5),quality(6),flexibility(6),in([[16,6],[49,18],[58,1]]),out([[26,8],
[38,16]]),mutual_exclusions([])).
task(id(70),cost(16),revenue(13),duration(7),quality(6),flexibility(0),in([[6,14],
[21,6]]),out([[16,4]]),mutual_exclusions([])).
task(id(71),cost(15),revenue(12),duration(9),quality(6),flexibility(6),in([[13,3],[69,3]]),out([[12,10],
[34,3]]),mutual_exclusions([])).
task(id(72),cost(13),revenue(10),duration(9),quality(5),flexibility(6),in([[2,14],[34,13],[41,14],
[68,14]]),out([[29,17]]),mutual_exclusions([])).
task(id(73),cost(6),revenue(5),duration(9),quality(3),flexibility(6),in([[25,9],[42,12],[49,5],[65,11]]),out([[4,14],[16,7],
[71,10]]),mutual_exclusions([])).
task(id(74),cost(7),revenue(6),duration(4),quality(7),flexibility(4),in([[26,19],[48,15],[60,4],[63,15]]),out([[25,8],
[50,16],[64,9]]),mutual_exclusions([])).
task(id(75),cost(15),revenue(9),duration(4),quality(4),flexibility(4),in([[28,19],
[46,2]]),out([[13,9]]),mutual_exclusions([])).
task(id(76),cost(14),revenue(8),duration(13),quality(2),flexibility(4),in([[3,15],[6,4],[20,12],[67,1],
[70,13]]),out([[19,4],[45,1],[50,2]]),mutual_exclusions([])).
task(id(77),cost(16),revenue(13),duration(7),quality(3),flexibility(3),in([[13,11],[21,15],[44,15],[50,6],
[67,13]]),out([[13,18],[29,13],[48,19]]),mutual_exclusions([])).
task(id(78),cost(15),revenue(12),duration(6),quality(7),flexibility(1),in([[11,5],[13,13],[72,3],[74,1]]),out([[3,9],[7,4],
[57,9]]),mutual_exclusions([])).
task(id(79),cost(17),revenue(11),duration(9),quality(0),flexibility(0),in([[13,16],[15,19],[29,12],[43,11],
[62,14]]),out([[12,19],[72,4]]),mutual_exclusions([])).
task(id(80),cost(7),revenue(6),duration(13),quality(4),flexibility(2),in([[1,11],[13,7],[18,7],[63,3]]),out([[20,3],[35,6],
[42,17]]),mutual_exclusions([])).
task(id(81),cost(13),revenue(10),duration(8),quality(1),flexibility(0),in([[21,8],
[58,10]]),out([[47,12]]),mutual_exclusions([])).
task(id(82),cost(13),revenue(7),duration(7),quality(2),flexibility(2),in([[5,6],[7,15],[25,14]]),out([[16,1],
[45,14]]),mutual_exclusions([])).
task(id(83),cost(16),revenue(10),duration(12),quality(7),flexibility(3),in([[16,2],[54,5],[59,14]]),out([[7,6],
[28,10]]),mutual_exclusions([])).
```

```
task(id(84),cost(7),revenue(3),duration(5),quality(1),flexibility(4),in([[18,16],[47,12],[54,13]]),out([[7,18],[24,5],
[66,3]]),mutual_exclusions([])).
task(id(85),cost(15),revenue(9),duration(10),quality(1),flexibility(6),in([[24,7],
[35,9]]),out([[2,15]]),mutual_exclusions([])).
task(id(86),cost(6),revenue(5),duration(12),quality(3),flexibility(0),in([[9,5],[13,17],[54,10]]),out([[38,2],[61,13],
[72,19]]),mutual_exclusions([])).
task(id(87),cost(16),revenue(10),duration(5),quality(3),flexibility(3),in([[30,14],[31,13],[41,7],[56,6],
[69,17]]),out([[2,11],[12,2],[42,13]]),mutual_exclusions([])).
task(id(88),cost(15),revenue(12),duration(10),quality(7),flexibility(0),in([[13,19],[16,16],[22,14],[45,4],
[50,3]]),out([[61,16],[67,19]]),mutual_exclusions([])).
task(id(89),cost(12),revenue(7),duration(4),quality(6),flexibility(7),in([[19,5],[50,15],[61,13]]),out([[18,6],
[37,4]]),mutual_exclusions([])).
task(id(90),cost(10),revenue(8),duration(9),quality(2),flexibility(5),in([[48,18],[65,12]]),out([[12,6],
[45,13]]),mutual_exclusions([])).
task(id(91),cost(13),revenue(10),duration(10),quality(0),flexibility(1),in([[6,1],[8,12],[63,3]]),out([[29,17],
[54,18]]),mutual_exclusions([])).
task(id(92),cost(11),revenue(9),duration(5),quality(0),flexibility(2),in([[28,6],[52,4],
[74,17]]),out([[18,19]]),mutual_exclusions([])).
task(id(93),cost(15),revenue(9),duration(6),quality(7),flexibility(1),in([[9,11],[36,11],[42,5],[51,19],
[66,17]]),out([[11,15],[15,4],[73,11]]),mutual_exclusions([])).
task(id(94),cost(13),revenue(10),duration(13),quality(2),flexibility(6),in([[2,6],[12,12],[23,4],[42,17],
[74,10]]),out([[23,12],[69,4]]),mutual_exclusions([])).
task(id(95),cost(15),revenue(9),duration(10),quality(2),flexibility(2),in([[3,12],[46,15],[52,6]]),out([[38,17],
[68,11]]),mutual_exclusions([])).
task(id(96),cost(15),revenue(9),duration(8),quality(0),flexibility(5),in([[3,11],[38,19],[53,7],[55,8]]),out([[29,10],
[43,2],[62,2]]),mutual_exclusions([])).
task(id(97),cost(10),revenue(5),duration(7),quality(4),flexibility(2),in([[16,5],[39,9],[55,5]]),out([[2,18],[45,8],
[59,11]]),mutual_exclusions([])).
task(id(98),cost(8),revenue(3),duration(9),quality(5),flexibility(3),in([[4,2],[20,13],[36,11]]),out([[33,2],[59,17],
[63,4]]),mutual_exclusions([])).
task(id(99),cost(14),revenue(8),duration(4),quality(1),flexibility(2),in([[20,11],[51,10],[52,7],[66,1]]),out([[8,19],
[50,19],[51,4]]),mutual_exclusions([])).
task(id(100),cost(17),revenue(14),duration(12),quality(3),flexibility(3),in([[3,6],[7,18],[20,3],[53,3],
[73,14]]),out([[1,14],[51,3],[55,3]]),mutual_exclusions([])).
task(id(101),cost(12),revenue(10),duration(10),quality(5),flexibility(5),in([[2,9],[12,8],[31,4],
[72,17]]),out([[21,18]]),mutual_exclusions([])).
task(id(102),cost(14),revenue(8),duration(12),quality(4),flexibility(3),in([[32,16],
[42,11]]),out([[25,18]]),mutual_exclusions([])).
task(id(103),cost(11),revenue(6),duration(7),quality(1),flexibility(0),in([[7,13],[18,4],[25,15],[54,5],
[72,10]]),out([[8,2]]),mutual_exclusions([])).
task(id(104),cost(9),revenue(4),duration(4),quality(2),flexibility(6),in([[10,5],[48,17],[64,1]]),out([[25,14],[27,12],
[63,2]]),mutual_exclusions([])).
task(id(105),cost(7),revenue(3),duration(12),quality(1),flexibility(0),in([[30,8],[47,7],[48,19]]),out([[4,14],[41,13],
[73,2]]),mutual_exclusions([])).
```

```
task(id(106),cost(7),revenue(3),duration(4),quality(4),flexibility(6),in([[4,1],[17,13],[28,3],
[71,3]]),out([[62,16]]),mutual_exclusions([])).
task(id(107),cost(8),revenue(6),duration(10),quality(0),flexibility(7),in([[7,3],[16,11]]),out([[23,14],
[72,17]]),mutual_exclusions([])).
task(id(108),cost(11),revenue(9),duration(5),quality(2),flexibility(7),in([[25,12],[38,1],[53,3],[57,11],
[65,12]]),out([[65,14]]),mutual_exclusions([])).
task(id(109),cost(8),revenue(6),duration(8),quality(0),flexibility(0),in([[6,3],[22,8],[40,17]]),out([[4,10],
[29,11]]),mutual_exclusions([])).
task(id(110),cost(11),revenue(6),duration(7),quality(2),flexibility(5),in([[1,10],[20,6],[43,4],
[46,19]]),out([[46,15]]),mutual_exclusions([])).
task(id(111),cost(6),revenue(5),duration(6),quality(5),flexibility(5),in([[1,10],[51,17],
[54,8]]),out([[52,11]]),mutual_exclusions([])).
task(id(112),cost(8),revenue(3),duration(6),quality(2),flexibility(1),in([[4,7],[9,17],[22,10],[31,17],
[69,11]]),out([[19,5]]),mutual_exclusions([])).
task(id(113),cost(7),revenue(6),duration(7),quality(6),flexibility(1),in([[18,18],[36,5],[60,14],[62,16]]),out([[1,8],
[20,16],[40,12]]),mutual_exclusions([])).
task(id(114),cost(16),revenue(10),duration(7),quality(6),flexibility(0),in([[46,14],[48,5],
[56,12]]),out([[69,7]]),mutual_exclusions([])).
task(id(115),cost(12),revenue(7),duration(12),quality(0),flexibility(5),in([[26,11],[56,8],
[58,14]]),out([[21,18]]),mutual_exclusions([])).
task(id(116),cost(14),revenue(11),duration(9),quality(2),flexibility(4),in([[42,8],[61,4]]),out([[17,13],[22,2],
[50,6]]),mutual_exclusions([])).
task(id(117),cost(6),revenue(5),duration(12),quality(2),flexibility(2),in([[34,1],[59,12]]),out([[14,19],
[74,18]]),mutual_exclusions([])).
task(id(118),cost(6),revenue(2),duration(8),quality(2),flexibility(1),in([[7,13],[8,3],[47,9],[63,2]]),out([[1,6],[6,7],
[16,17]]),mutual_exclusions([])).
task(id(119),cost(11),revenue(9),duration(12),quality(3),flexibility(2),in([[58,12],[60,1]]),out([[31,16],
[39,19]]),mutual_exclusions([])).
task(id(120),cost(14),revenue(11),duration(12),quality(5),flexibility(5),in([[36,13],[42,4],[62,15],[69,5],
[70,17]]),out([[10,1],[59,17]]),mutual_exclusions([])).
task(id(121),cost(6),revenue(5),duration(12),quality(3),flexibility(2),in([[5,16],[21,5],[35,16],
[57,7]]),out([[48,13]]),mutual_exclusions([])).
task(id(122),cost(8),revenue(6),duration(9),quality(1),flexibility(4),in([[29,7],[63,2],[72,4]]),out([[9,19],
[63,8]]),mutual_exclusions([])).
task(id(123),cost(16),revenue(13),duration(4),quality(3),flexibility(6),in([[7,14],[32,6],[33,18]]),out([[28,9],[65,1],
[66,2]]),mutual_exclusions([])).
task(id(124),cost(10),revenue(5),duration(8),quality(3),flexibility(3),in([[17,14],[26,12],[32,5]]),out([[7,9],
[43,1]]),mutual_exclusions([])).
task(id(125),cost(9),revenue(7),duration(7),quality(6),flexibility(5),in([[10,9],
[32,17]]),out([[3,1]]),mutual_exclusions([])).
task(id(126),cost(12),revenue(10),duration(6),quality(2),flexibility(3),in([[4,7],[25,19],[56,11],[68,11],
[70,7]]),out([[2,6],[4,6],[49,5]]),mutual_exclusions([])).
task(id(127),cost(6),revenue(5),duration(11),quality(0),flexibility(3),in([[60,16],[71,7]]),out([[5,10],
[8,1]]),mutual_exclusions([])).
```

```
task(id(128),cost(17),revenue(11),duration(13),quality(7),flexibility(0),in([[8,2],[19,19],[36,7],[53,9],
[70,12]]),out([[6,15]]),mutual_exclusions([])).
task(id(129),cost(11),revenue(6),duration(6),quality(3),flexibility(0),in([[16,3],[27,2],[43,14]]),out([[7,16],[23,15],
[36,5]]),mutual_exclusions([])).
task(id(130),cost(9),revenue(7),duration(5),quality(1),flexibility(4),in([[13,9],[16,1]]),out([[1,14],[2,19],
[14,5]]),mutual_exclusions([])).
task(id(131),cost(7),revenue(3),duration(11),quality(7),flexibility(6),in([[2,7],[11,19],[17,7],[43,2],
[71,7]]),out([[23,13]]),mutual_exclusions([])).
task(id(132),cost(11),revenue(6),duration(12),quality(0),flexibility(5),in([[27,7],[33,1],[39,14],[70,13],
[73,17]]),out([[9,8],[25,15],[42,16]]),mutual_exclusions([])).
task(id(133),cost(8),revenue(3),duration(11),quality(7),flexibility(6),in([[25,2],[40,2],[55,6]]),out([[15,10],
[64,18]]),mutual_exclusions([])).
task(id(134),cost(9),revenue(4),duration(9),quality(0),flexibility(3),in([[9,15],[31,2]]),out([[37,17],
[54,18]]),mutual_exclusions([])).
task(id(135),cost(10),revenue(5),duration(9),quality(3),flexibility(1),in([[12,4],[51,16],[67,18]]),out([[15,14],[25,15],
[73,5]]),mutual_exclusions([])).
task(id(136),cost(13),revenue(10),duration(4),quality(4),flexibility(1),in([[9,7],[23,10],[30,1],[61,5]]),out([[34,1],
[65,4],[68,5]]),mutual_exclusions([])).
task(id(137),cost(7),revenue(3),duration(5),quality(6),flexibility(6),in([[21,11],[45,3]]),out([[5,7],
[50,13]]),mutual_exclusions([])).
task(id(138),cost(9),revenue(4),duration(9),quality(3),flexibility(6),in([[23,4],[48,1],[64,16],[68,18]]),out([[6,6],
[71,11]]),mutual_exclusions([])).
task(id(139),cost(16),revenue(10),duration(9),quality(3),flexibility(4),in([[12,16],[43,8],[48,9],[57,1],
[73,18]]),out([[1,19],[45,10],[48,3]]),mutual_exclusions([])).
task(id(140),cost(16),revenue(13),duration(6),quality(6),flexibility(1),in([[51,6],[68,4]]),out([[20,1],
[48,10]]),mutual_exclusions([])).
task(id(141),cost(17),revenue(14),duration(13),quality(5),flexibility(1),in([[3,17],[56,2]]),out([[43,12],[52,2],
[57,8]]),mutual_exclusions([])).
task(id(142),cost(13),revenue(10),duration(10),quality(2),flexibility(5),in([[8,1],[24,9],[52,5],[68,6]]),out([[15,12],
[58,18],[60,13]]),mutual_exclusions([])).
task(id(143),cost(8),revenue(6),duration(6),quality(2),flexibility(5),in([[20,9],[31,11],[45,4],[48,8],
[67,19]]),out([[12,4],[45,6]]),mutual_exclusions([])).
task(id(144),cost(11),revenue(9),duration(6),quality(5),flexibility(4),in([[33,9],[38,11],[44,18],[71,15],
[72,9]]),out([[2,15],[37,14],[51,12]]),mutual_exclusions([])).
task(id(145),cost(8),revenue(3),duration(4),quality(0),flexibility(0),in([[20,8],[59,5],
[69,17]]),out([[69,11]]),mutual_exclusions([])).
task(id(146),cost(12),revenue(7),duration(7),quality(4),flexibility(5),in([[8,16],[10,19],[31,7]]),out([[5,1],[18,7],
[63,11]]),mutual_exclusions([])).
task(id(147),cost(15),revenue(9),duration(11),quality(7),flexibility(6),in([[29,7],[42,13],[53,6],[69,7]]),out([[39,5],
[41,15]]),mutual_exclusions([])).
task(id(148),cost(16),revenue(10),duration(7),quality(7),flexibility(1),in([[11,2],[37,1],[47,7],[49,6]]),out([[19,18],
[20,19],[40,9]]),mutual_exclusions([])).
task(id(149),cost(8),revenue(6),duration(13),quality(5),flexibility(1),in([[9,4],[37,13],[53,5],[56,13]]),out([[4,16],
[9,15],[73,16]]),mutual_exclusions([])).
```

task(id(150),cost(9),revenue(7),duration(13),quality(1),flexibility(7),in([[29,2],
[38,13]]),out([[4,10]]),mutual_exclusions([])).
task(id(151),cost(16),revenue(13),duration(9),quality(1),flexibility(6),in([[17,13],[37,19],[38,2],[62,9]]),out([[11,18],
[45,19],[65,12]]),mutual_exclusions([])).
task(id(152),cost(8),revenue(6),duration(11),quality(6),flexibility(2),in([[11,8],[57,9]]),out([[48,7],
[54,1]]),mutual_exclusions([])).
task(id(153),cost(17),revenue(11),duration(11),quality(7),flexibility(7),in([[8,1],[25,13],[36,10],[65,18],
[66,14]]),out([[19,5],[36,14],[74,17]]),mutual_exclusions([])).
task(id(154),cost(6),revenue(2),duration(4),quality(0),flexibility(1),in([[15,11],[33,9]]),out([[36,9],
[60,8]]),mutual_exclusions([])).
task(id(155),cost(17),revenue(11),duration(12),quality(7),flexibility(4),in([[34,4],[72,9]]),out([[3,4],
[72,3]]),mutual_exclusions([])).
task(id(156),cost(7),revenue(6),duration(5),quality(1),flexibility(2),in([[20,4],[34,13],[45,14],[59,13],
[62,12]]),out([[52,19]]),mutual_exclusions([])).
task(id(157),cost(11),revenue(6),duration(5),quality(0),flexibility(1),in([[1,18],[4,2],[7,7],[24,11]]),out([[7,18],
[73,14]]),mutual_exclusions([])).
task(id(158),cost(6),revenue(2),duration(9),quality(4),flexibility(5),in([[3,10],[17,17]]),out([[11,18],[37,16],
[57,3]]),mutual_exclusions([])).
task(id(159),cost(9),revenue(4),duration(4),quality(5),flexibility(0),in([[9,8],[22,2],[30,16],[31,17],
[44,5]]),out([[24,7]]),mutual_exclusions([])).
task(id(160),cost(6),revenue(2),duration(12),quality(5),flexibility(7),in([[33,10],[34,7],[44,3],[56,2]]),out([[4,14],
[11,15],[63,16]]),mutual_exclusions([])).
task(id(161),cost(10),revenue(8),duration(5),quality(6),flexibility(1),in([[3,12],[4,13],
[51,18]]),out([[33,5]]),mutual_exclusions([])).
task(id(162),cost(6),revenue(2),duration(8),quality(1),flexibility(3),in([[14,16],[50,5],[52,12]]),out([[5,16],
[60,17]]),mutual_exclusions([])).
task(id(163),cost(15),revenue(9),duration(7),quality(7),flexibility(7),in([[9,15],[16,11],[21,1],[25,18]]),out([[2,18],
[8,15],[73,2]]),mutual_exclusions([])).
task(id(164),cost(17),revenue(14),duration(8),quality(0),flexibility(4),in([[13,2],[26,5],[40,2],[47,9],
[65,17]]),out([[4,6],[14,13]]),mutual_exclusions([])).
task(id(165),cost(12),revenue(7),duration(6),quality(5),flexibility(3),in([[33,18],[34,14],
[44,18]]),out([[74,18]]),mutual_exclusions([])).
task(id(166),cost(9),revenue(4),duration(8),quality(7),flexibility(4),in([[51,3],
[55,15]]),out([[30,18]]),mutual_exclusions([])).
task(id(167),cost(7),revenue(3),duration(10),quality(6),flexibility(2),in([[7,12],[29,1],[35,3],[61,13]]),out([[28,6],
[29,6]]),mutual_exclusions([])).
task(id(168),cost(9),revenue(7),duration(9),quality(6),flexibility(1),in([[1,6],[16,5],[19,14],
[72,3]]),out([[29,3]]),mutual_exclusions([])).
task(id(169),cost(14),revenue(8),duration(13),quality(2),flexibility(0),in([[44,5],[68,1]]),out([[9,13],[34,10],
[68,17]]),mutual_exclusions([])).
task(id(170),cost(16),revenue(13),duration(5),quality(4),flexibility(0),in([[39,3],[54,14],[62,8],[71,5],
[73,2]]),out([[36,4],[37,16],[58,7]]),mutual_exclusions([])).
task(id(171),cost(7),revenue(6),duration(7),quality(7),flexibility(3),in([[50,5],[60,11]]),out([[15,4],[25,5],
[32,11]]),mutual_exclusions([])).

```
task(id(172),cost(14),revenue(8),duration(6),quality(7),flexibility(7),in([[9,11],[12,1],[43,3]]),out([[33,13],[34,2],
[64,2]]),mutual_exclusions([])).
task(id(173),cost(12),revenue(10),duration(7),quality(0),flexibility(7),in([[6,16],[11,4],
[13,13]]),out([[32,1]]),mutual_exclusions([])).
task(id(174),cost(13),revenue(7),duration(6),quality(1),flexibility(0),in([[3,9],[58,6]]),out([[10,17],
[55,9]]),mutual_exclusions([])).
task(id(175),cost(11),revenue(6),duration(10),quality(0),flexibility(2),in([[14,17],[16,8],
[26,8]]),out([[67,11]]),mutual_exclusions([])).
task(id(176),cost(11),revenue(6),duration(9),quality(2),flexibility(0),in([[4,19],[18,7],[48,1]]),out([[18,5],
[21,6]]),mutual_exclusions([])).
task(id(177),cost(6),revenue(5),duration(8),quality(2),flexibility(6),in([[13,14],[34,17]]),out([[10,6],
[56,12]]),mutual_exclusions([])).
task(id(178),cost(11),revenue(6),duration(4),quality(6),flexibility(6),in([[43,18],[55,15],
[74,8]]),out([[13,14]]),mutual_exclusions([])).
task(id(179),cost(7),revenue(3),duration(11),quality(2),flexibility(0),in([[37,14],[39,8],[48,15],[56,13],
[72,8]]),out([[6,7],[47,17],[51,10]]),mutual_exclusions([])).
task(id(180),cost(11),revenue(9),duration(10),quality(6),flexibility(7),in([[11,13],
[18,9]]),out([[13,3]]),mutual_exclusions([])).
task(id(181),cost(16),revenue(13),duration(9),quality(0),flexibility(2),in([[27,2],[28,19],[38,3],[56,4],
[59,17]]),out([[7,8],[58,15]]),mutual_exclusions([])).
task(id(182),cost(15),revenue(12),duration(9),quality(0),flexibility(6),in([[2,13],[16,9],[26,14],[32,2],
[64,6]]),out([[61,9]]),mutual_exclusions([])).
task(id(183),cost(8),revenue(6),duration(9),quality(5),flexibility(2),in([[2,6],[25,12]]),out([[40,19],
[58,1]]),mutual_exclusions([])).
task(id(184),cost(12),revenue(10),duration(4),quality(7),flexibility(7),in([[20,10],[29,8],[46,13],[65,17],
[73,11]]),out([[7,1],[39,4],[54,9]]),mutual_exclusions([])).
task(id(185),cost(8),revenue(6),duration(13),quality(4),flexibility(0),in([[5,13],[7,4]]),out([[7,6],[12,8],
[62,13]]),mutual_exclusions([])).
task(id(186),cost(15),revenue(9),duration(7),quality(0),flexibility(2),in([[29,13],[30,8]]),out([[12,4],[31,9],
[38,19]]),mutual_exclusions([])).
task(id(187),cost(11),revenue(9),duration(10),quality(2),flexibility(1),in([[17,11],[18,9],[37,3],
[54,4]]),out([[69,15]]),mutual_exclusions([])).
task(id(188),cost(15),revenue(12),duration(12),quality(5),flexibility(3),in([[33,2],[34,10],[43,1],[57,18],
[62,8]]),out([[69,5],[74,12]]),mutual_exclusions([])).
task(id(189),cost(7),revenue(3),duration(12),quality(0),flexibility(2),in([[44,19],[54,13],[57,14]]),out([[26,16],[50,4],
[74,1]]),mutual_exclusions([])).
task(id(190),cost(13),revenue(7),duration(10),quality(6),flexibility(2),in([[25,17],[26,15],[45,17],
[51,19]]),out([[54,19]]),mutual_exclusions([])).
task(id(191),cost(10),revenue(8),duration(6),quality(2),flexibility(7),in([[7,18],[17,3],[35,10],[38,10],
[52,7]]),out([[65,5]]),mutual_exclusions([])).
task(id(192),cost(10),revenue(5),duration(10),quality(7),flexibility(0),in([[7,6],[22,7],[40,6],[51,13],
[70,17]]),out([[8,18]]),mutual_exclusions([])).
task(id(193),cost(6),revenue(2),duration(5),quality(7),flexibility(7),in([[37,2],[51,10],
[68,14]]),out([[29,4]]),mutual_exclusions([])).
```

```
task(id(194),cost(12),revenue(10),duration(10),quality(4),flexibility(7),in([[1,16],[16,16],[59,5],
[70,5]]),out([[38,10]]),mutual_exclusions([])).
task(id(195),cost(12),revenue(10),duration(4),quality(4),flexibility(7),in([[8,7],[27,16],[44,17],[51,19]]),out([[17,1],
[33,1]]),mutual_exclusions([])).
task(id(196),cost(10),revenue(5),duration(5),quality(0),flexibility(4),in([[4,16],
[51,10]]),out([[64,8]]),mutual_exclusions([])).
task(id(197),cost(9),revenue(7),duration(8),quality(7),flexibility(6),in([[20,5],[36,6]]),out([[49,3],[66,14],
[73,2]]),mutual_exclusions([])).
task(id(198),cost(10),revenue(8),duration(7),quality(2),flexibility(2),in([[5,4],
[61,13]]),out([[47,6]]),mutual_exclusions([])).
task(id(199),cost(15),revenue(12),duration(12),quality(3),flexibility(5),in([[2,9],[39,18],[58,10]]),out([[17,1],[44,8],
[55,2]]),mutual_exclusions([])).
task(id(200),cost(6),revenue(2),duration(12),quality(3),flexibility(3),in([[13,17],[35,18],[54,16],[64,2],
[68,16]]),out([[2,9]]),mutual_exclusions([])).
task(id(201),cost(11),revenue(6),duration(7),quality(6),flexibility(1),in([[7,11],[11,3],[34,14],[47,14]]),out([[2,7],
[20,16],[27,7]]),mutual_exclusions([])).
task(id(202),cost(17),revenue(11),duration(13),quality(7),flexibility(7),in([[6,14],[30,17],[49,10],[61,13],
[67,8]]),out([[7,7],[63,8],[65,11]]),mutual_exclusions([])).
task(id(203),cost(16),revenue(13),duration(10),quality(4),flexibility(3),in([[8,11],[20,11],[38,7],[58,6]]),out([[16,12],
[20,11],[73,7]]),mutual_exclusions([])).
task(id(204),cost(17),revenue(14),duration(5),quality(4),flexibility(0),in([[45,5],
[53,18]]),out([[74,2]]),mutual_exclusions([])).
task(id(205),cost(11),revenue(9),duration(10),quality(3),flexibility(7),in([[26,8],[54,7],[58,12],[62,18]]),out([[25,8],
[54,18]]),mutual_exclusions([])).
task(id(206),cost(11),revenue(9),duration(10),quality(0),flexibility(7),in([[5,8],[24,17],[25,6],[52,7]]),out([[50,19],
[66,9]]),mutual_exclusions([])).
task(id(207),cost(16),revenue(10),duration(10),quality(3),flexibility(7),in([[26,3],[30,7],[34,3],[57,13]]),out([[8,7],
[56,3],[73,19]]),mutual_exclusions([])).
task(id(208),cost(12),revenue(7),duration(13),quality(5),flexibility(6),in([[26,8],[63,9]]),out([[42,1],
[71,18]]),mutual_exclusions([])).
task(id(209),cost(14),revenue(11),duration(10),quality(0),flexibility(0),in([[51,15],[57,1],[59,11],[73,1]]),out([[22,17],
[27,6]]),mutual_exclusions([])).
task(id(210),cost(13),revenue(10),duration(6),quality(3),flexibility(2),in([[19,5],[26,17],[27,15],[67,2]]),out([[7,16],
[21,2]]),mutual_exclusions([])).
task(id(211),cost(7),revenue(6),duration(10),quality(4),flexibility(4),in([[19,13],
[35,2]]),out([[74,6]]),mutual_exclusions([])).
task(id(212),cost(11),revenue(9),duration(6),quality(0),flexibility(7),in([[17,14],[19,3],[30,12],[73,2]]),out([[4,7],
[42,11]]),mutual_exclusions([])).
task(id(213),cost(6),revenue(5),duration(13),quality(0),flexibility(4),in([[1,12],[45,15],[62,5]]),out([[23,9],
[31,14]]),mutual_exclusions([])).
task(id(214),cost(16),revenue(13),duration(12),quality(0),flexibility(4),in([[13,17],[19,3],[37,4],[44,1],
[71,13]]),out([[6,16],[60,3],[69,11]]),mutual_exclusions([])).
task(id(215),cost(7),revenue(3),duration(5),quality(0),flexibility(5),in([[12,19],[22,18],[40,10],[69,6]]),out([[27,5],
[68,14]]),mutual_exclusions([])).
```

```
task(id(216),cost(6),revenue(2),duration(11),quality(1),flexibility(0),in([[5,6],[40,9],[41,8],[71,4]]),out([[15,6],[29,10],
[47,16]]),mutual_exclusions([])).
task(id(217),cost(12),revenue(10),duration(10),quality(4),flexibility(1),in([[6,19],[8,12],[10,11],[61,2],
[72,12]]),out([[11,16],[14,8],[48,7]]),mutual_exclusions([])).
task(id(218),cost(8),revenue(3),duration(11),quality(5),flexibility(6),in([[21,7],[40,2],[50,19],[61,14]]),out([[45,14],
[52,10],[66,18]]),mutual_exclusions([])).
task(id(219),cost(6),revenue(5),duration(10),quality(7),flexibility(4),in([[3,16],[20,5],[39,7]]),out([[50,3],
[68,19]]),mutual_exclusions([])).
task(id(220),cost(14),revenue(8),duration(9),quality(3),flexibility(2),in([[22,2],[26,3],[28,7],[34,5],
[70,17]]),out([[25,9],[31,5],[46,15]]),mutual_exclusions([])).
task(id(221),cost(15),revenue(9),duration(9),quality(1),flexibility(4),in([[18,2],[20,12],[56,15]]),out([[1,6],[17,5],
[24,8]]),mutual_exclusions([])).
task(id(222),cost(7),revenue(6),duration(9),quality(2),flexibility(4),in([[23,8],[55,15]]),out([[14,16],[53,9],
[70,10]]),mutual_exclusions([])).
task(id(223),cost(15),revenue(9),duration(12),quality(6),flexibility(1),in([[10,1],[21,8],[61,5],[63,8]]),out([[21,18],
[42,18]]),mutual_exclusions([])).
task(id(224),cost(14),revenue(11),duration(7),quality(1),flexibility(5),in([[11,11],[43,18]]),out([[40,18],[41,17],
[51,12]]),mutual_exclusions([])).
task(id(225),cost(8),revenue(6),duration(9),quality(4),flexibility(4),in([[18,7],[69,17],
[74,7]]),out([[65,1]]),mutual_exclusions([])).
task(id(226),cost(11),revenue(9),duration(12),quality(1),flexibility(7),in([[6,2],[31,15],[37,17],[52,11]]),out([[16,16],
[31,5]]),mutual_exclusions([])).
task(id(227),cost(15),revenue(12),duration(11),quality(3),flexibility(4),in([[6,4],[18,8],[24,4],[40,9],
[72,11]]),out([[61,11]]),mutual_exclusions([])).
task(id(228),cost(7),revenue(6),duration(6),quality(7),flexibility(6),in([[25,19],
[32,5]]),out([[71,12]]),mutual_exclusions([])).
task(id(229),cost(17),revenue(11),duration(8),quality(7),flexibility(2),in([[19,5],[26,4],[69,9]]),out([[7,4],[15,12],
[20,8]]),mutual_exclusions([])).
task(id(230),cost(12),revenue(7),duration(4),quality(4),flexibility(3),in([[16,13],[34,17],[36,6]]),out([[11,12],[55,10],
[67,12]]),mutual_exclusions([])).
task(id(231),cost(13),revenue(7),duration(4),quality(5),flexibility(2),in([[32,13],
[72,13]]),out([[43,7]]),mutual_exclusions([])).
task(id(232),cost(11),revenue(9),duration(9),quality(0),flexibility(6),in([[2,6],[24,16],
[46,5]]),out([[59,13]]),mutual_exclusions([])).
task(id(233),cost(10),revenue(8),duration(13),quality(5),flexibility(1),in([[2,15],[15,12],[37,7],[41,5],
[48,1]]),out([[13,3],[14,10],[28,17]]),mutual_exclusions([])).
task(id(234),cost(15),revenue(12),duration(5),quality(1),flexibility(3),in([[17,4],[25,6],
[33,10]]),out([[4,18]]),mutual_exclusions([])).
task(id(235),cost(13),revenue(7),duration(7),quality(7),flexibility(7),in([[18,12],[48,13],[64,18]]),out([[5,14],
[15,18]]),mutual_exclusions([])).
task(id(236),cost(9),revenue(7),duration(12),quality(7),flexibility(0),in([[14,4],[55,19],[57,2],[61,15],
[71,7]]),out([[5,16]]),mutual_exclusions([])).
task(id(237),cost(8),revenue(6),duration(8),quality(6),flexibility(5),in([[4,16],[6,9],[25,4],[30,4],
[69,5]]),out([[47,13]]),mutual_exclusions([])).
```

```
task(id(238),cost(15),revenue(9),duration(4),quality(5),flexibility(7),in([[10,18],[22,6],[33,13],[64,8],
[72,9]]),out([[36,2]]),mutual_exclusions([])).
task(id(239),cost(15),revenue(9),duration(11),quality(5),flexibility(3),in([[42,17],[56,9]]),out([[20,2],[64,17],
[73,1]]),mutual_exclusions([])).
task(id(240),cost(11),revenue(9),duration(12),quality(1),flexibility(4),in([[12,12],[18,19],[36,3],[49,4]]),out([[22,12],
[28,18],[52,4]]),mutual_exclusions([])).
task(id(241),cost(7),revenue(6),duration(6),quality(1),flexibility(0),in([[22,4],[39,9],[54,1]]),out([[20,5],[31,10],
[54,6]]),mutual_exclusions([])).
task(id(242),cost(17),revenue(14),duration(7),quality(7),flexibility(0),in([[57,9],[67,15]]),out([[6,5],[21,10],
[51,12]]),mutual_exclusions([])).
task(id(243),cost(14),revenue(8),duration(6),quality(5),flexibility(0),in([[1,9],[38,8]]),out([[40,4],
[74,19]]),mutual_exclusions([])).
task(id(244),cost(14),revenue(8),duration(9),quality(1),flexibility(0),in([[18,17],
[46,18]]),out([[5,6]]),mutual_exclusions([])).
task(id(245),cost(6),revenue(5),duration(6),quality(3),flexibility(2),in([[30,15],[52,15],
[53,19]]),out([[30,15]]),mutual_exclusions([])).
task(id(246),cost(17),revenue(11),duration(6),quality(3),flexibility(3),in([[19,9],[20,6],
[24,1]]),out([[27,15]]),mutual_exclusions([])).
task(id(247),cost(8),revenue(3),duration(12),quality(4),flexibility(0),in([[16,10],[38,13],
[40,7]]),out([[11,15]]),mutual_exclusions([])).
task(id(248),cost(7),revenue(3),duration(10),quality(7),flexibility(3),in([[24,6],[28,6],[44,2],[50,3],[52,9]]),out([[7,13],
[47,19],[53,2]]),mutual_exclusions([])).
task(id(249),cost(17),revenue(14),duration(8),quality(5),flexibility(2),in([[13,12],[18,7],[26,11],[38,14],
[69,15]]),out([[63,5]]),mutual_exclusions([])).
task(id(250),cost(17),revenue(14),duration(6),quality(0),flexibility(5),in([[34,3],[35,7],[66,14]]),out([[8,3],[36,5],
[61,10]]),mutual_exclusions([])).
task(id(251),cost(9),revenue(7),duration(5),quality(7),flexibility(6),in([[33,11],[51,18],
[59,14]]),out([[12,1]]),mutual_exclusions([])).
task(id(252),cost(15),revenue(12),duration(13),quality(1),flexibility(6),in([[14,6],
[63,9]]),out([[62,19]]),mutual_exclusions([])).
task(id(253),cost(14),revenue(8),duration(6),quality(4),flexibility(7),in([[33,10],
[56,11]]),out([[8,18]]),mutual_exclusions([])).
task(id(254),cost(6),revenue(2),duration(5),quality(3),flexibility(2),in([[1,15],[17,10],[27,10],[34,19]]),out([[9,15],
[24,10],[71,10]]),mutual_exclusions([])).
task(id(255),cost(12),revenue(10),duration(8),quality(6),flexibility(7),in([[11,13],[31,17],[65,16],[69,17]]),out([[39,7],
[65,14],[67,19]]),mutual_exclusions([])).
task(id(256),cost(12),revenue(10),duration(11),quality(1),flexibility(7),in([[4,4],[9,1],[24,10],[61,2],
[69,13]]),out([[4,18],[56,7]]),mutual_exclusions([])).
task(id(257),cost(16),revenue(13),duration(8),quality(0),flexibility(4),in([[18,1],[26,7],[27,1],[40,13],
[58,1]]),out([[53,19],[59,4]]),mutual_exclusions([])).
task(id(258),cost(10),revenue(8),duration(12),quality(1),flexibility(3),in([[4,9],[51,1]]),out([[50,4],
[69,7]]),mutual_exclusions([])).
task(id(259),cost(15),revenue(9),duration(8),quality(1),flexibility(2),in([[38,13],[44,2],[62,6],
[65,13]]),out([[23,19]]),mutual_exclusions([])).
```

task(id(260),cost(12),revenue(10),duration(9),quality(1),flexibility(5),in([[10,4],[64,1]]),out([[13,1],
[15,19]]),mutual_exclusions([])).
task(id(261),cost(10),revenue(5),duration(11),quality(3),flexibility(1),in([[28,17],[36,1],[70,10],
[74,6]]),out([[20,4]]),mutual_exclusions([])).
task(id(262),cost(11),revenue(9),duration(13),quality(5),flexibility(7),in([[2,17],[3,3],
[4,17]]),out([[73,18]]),mutual_exclusions([])).
task(id(263),cost(6),revenue(2),duration(7),quality(6),flexibility(4),in([[14,3],[16,15],[53,15],[55,18],
[72,1]]),out([[50,17]]),mutual_exclusions([])).
task(id(264),cost(8),revenue(3),duration(12),quality(4),flexibility(2),in([[49,1],[53,10],
[62,14]]),out([[41,9]]),mutual_exclusions([])).
task(id(265),cost(13),revenue(10),duration(8),quality(3),flexibility(4),in([[8,18],[43,2],[58,13],[68,2]]),out([[41,2],
[70,12]]),mutual_exclusions([])).
task(id(266),cost(15),revenue(9),duration(13),quality(0),flexibility(7),in([[6,4],[31,8],[32,8],[49,1]]),out([[29,7],
[64,18],[68,13]]),mutual_exclusions([])).
task(id(267),cost(8),revenue(6),duration(6),quality(5),flexibility(0),in([[31,2],[59,8]]),out([[19,7],
[55,5]]),mutual_exclusions([])).
task(id(268),cost(13),revenue(10),duration(7),quality(6),flexibility(7),in([[11,3],[21,9],[30,14],[63,3],
[71,12]]),out([[29,15],[60,19]]),mutual_exclusions([])).
task(id(269),cost(6),revenue(2),duration(13),quality(2),flexibility(0),in([[10,9],
[52,14]]),out([[50,6]]),mutual_exclusions([])).
task(id(270),cost(9),revenue(7),duration(5),quality(0),flexibility(4),in([[17,9],[27,9],[34,11],[35,5]]),out([[56,7],
[66,16]]),mutual_exclusions([])).
task(id(271),cost(14),revenue(8),duration(4),quality(0),flexibility(4),in([[21,7],[61,18]]),out([[33,4],
[67,9]]),mutual_exclusions([])).
task(id(272),cost(11),revenue(9),duration(10),quality(7),flexibility(3),in([[5,12],[9,4],
[53,17]]),out([[35,17]]),mutual_exclusions([])).
task(id(273),cost(15),revenue(9),duration(11),quality(2),flexibility(4),in([[50,12],[52,6],[54,7],[65,3]]),out([[14,15],
[40,5]]),mutual_exclusions([])).
task(id(274),cost(16),revenue(10),duration(8),quality(7),flexibility(6),in([[24,18],[30,17],[33,2],
[34,2]]),out([[32,12]]),mutual_exclusions([])).
task(id(275),cost(13),revenue(10),duration(13),quality(7),flexibility(7),in([[9,13],[23,13],[38,12],
[56,11]]),out([[11,16],[68,18]]),mutual_exclusions([])).
task(id(276),cost(14),revenue(11),duration(4),quality(3),flexibility(2),in([[29,5],[31,5],[48,3],[51,19],
[68,11]]),out([[27,18],[62,19]]),mutual_exclusions([])).
task(id(277),cost(16),revenue(10),duration(12),quality(1),flexibility(1),in([[18,11],[21,8],[22,10],[30,9]]),out([[13,19],
[23,5],[72,8]]),mutual_exclusions([])).
task(id(278),cost(16),revenue(10),duration(9),quality(2),flexibility(3),in([[1,9],[31,9],[33,8],[45,9]]),out([[29,16],
[41,8],[64,10]]),mutual_exclusions([])).
task(id(279),cost(11),revenue(9),duration(7),quality(7),flexibility(5),in([[22,4],[28,11],[40,15],[61,10],
[62,11]]),out([[20,13]]),mutual_exclusions([])).
task(id(280),cost(14),revenue(8),duration(8),quality(0),flexibility(4),in([[30,15],[35,2],[47,7],[58,11],
[70,14]]),out([[39,8],[56,11]]),mutual_exclusions([])).
task(id(281),cost(14),revenue(11),duration(7),quality(4),flexibility(7),in([[68,7],[69,15],
[74,18]]),out([[64,13]]),mutual_exclusions([])).

```
task(id(282),cost(13),revenue(10),duration(11),quality(2),flexibility(6),in([[48,14],[67,19]]),out([[8,8],[39,19],
[66,3]]),mutual_exclusions([])).
task(id(283),cost(9),revenue(7),duration(13),quality(4),flexibility(6),in([[15,6],[27,4],[71,16],[72,16],
[73,9]]),out([[28,8],[30,2],[59,17]]),mutual_exclusions([])).
task(id(284),cost(12),revenue(7),duration(5),quality(4),flexibility(6),in([[7,6],[17,12],[39,3]]),out([[17,18],
[71,2]]),mutual_exclusions([])).
task(id(285),cost(12),revenue(7),duration(11),quality(3),flexibility(3),in([[18,9],[24,4],[26,18],[73,18]]),out([[13,11],
[54,5]]),mutual_exclusions([])).
task(id(286),cost(16),revenue(10),duration(12),quality(4),flexibility(7),in([[3,5],[19,9],[29,2]]),out([[65,5],
[71,11]]),mutual_exclusions([])).
task(id(287),cost(8),revenue(3),duration(9),quality(5),flexibility(6),in([[5,8],[16,9],[60,15]]),out([[3,6],[8,8],
[60,4]]),mutual_exclusions([])).
task(id(288),cost(7),revenue(6),duration(7),quality(5),flexibility(1),in([[50,7],
[56,7]]),out([[62,18]]),mutual_exclusions([])).
task(id(289),cost(14),revenue(11),duration(10),quality(4),flexibility(3),in([[9,1],
[48,11]]),out([[49,9]]),mutual_exclusions([])).
task(id(290),cost(17),revenue(14),duration(11),quality(2),flexibility(2),in([[23,1],[67,14],
[70,16]]),out([[66,19]]),mutual_exclusions([])).
task(id(291),cost(10),revenue(8),duration(4),quality(3),flexibility(7),in([[48,9],[57,15]]),out([[18,6],[35,8],
[51,9]]),mutual_exclusions([])).
task(id(292),cost(17),revenue(14),duration(8),quality(0),flexibility(3),in([[10,5],[17,18],[29,18],
[51,1]]),out([[33,1]]),mutual_exclusions([])).
task(id(293),cost(11),revenue(9),duration(13),quality(3),flexibility(7),in([[19,14],[47,15],[50,15],[64,11],
[67,13]]),out([[21,11],[22,8]]),mutual_exclusions([])).
task(id(294),cost(7),revenue(6),duration(4),quality(1),flexibility(6),in([[21,6],[52,2],
[56,10]]),out([[65,12]]),mutual_exclusions([])).
task(id(295),cost(16),revenue(10),duration(6),quality(7),flexibility(4),in([[8,10],[11,15],[43,6],[61,18],
[74,4]]),out([[23,3]]),mutual_exclusions([])).
task(id(296),cost(17),revenue(14),duration(5),quality(6),flexibility(1),in([[59,10],[63,15]]),out([[2,10],[23,1],
[64,13]]),mutual_exclusions([])).
task(id(297),cost(14),revenue(8),duration(11),quality(6),flexibility(2),in([[17,7],[53,2],[56,2]]),out([[20,9],
[42,17]]),mutual_exclusions([])).
task(id(298),cost(16),revenue(10),duration(8),quality(1),flexibility(2),in([[9,13],[26,10],[43,5],
[60,8]]),out([[61,5]]),mutual_exclusions([])).
task(id(299),cost(15),revenue(9),duration(13),quality(7),flexibility(3),in([[13,5],[35,10],[56,9],[57,4],
[71,15]]),out([[4,9]]),mutual_exclusions([])).
task(id(300),cost(11),revenue(9),duration(5),quality(0),flexibility(6),in([[14,12],
[60,10]]),out([[26,8]]),mutual_exclusions([])).
task(id(301),cost(17),revenue(14),duration(7),quality(1),flexibility(4),in([[9,11],[10,14],[23,6],[30,7],
[67,15]]),out([[17,12],[20,5],[39,12]]),mutual_exclusions([])).
task(id(302),cost(17),revenue(14),duration(6),quality(2),flexibility(5),in([[11,19],[26,10],[43,6]]),out([[15,5],[24,15],
[66,18]]),mutual_exclusions([])).
task(id(303),cost(15),revenue(9),duration(4),quality(1),flexibility(1),in([[4,12],[7,9],[35,6],[56,8]]),out([[16,16],
[23,10]]),mutual_exclusions([])).
```

```
task(id(304),cost(8),revenue(6),duration(7),quality(1),flexibility(0),in([[33,13],[35,13],[53,15],
[57,11]]),out([[74,4]]),mutual_exclusions([])).
task(id(305),cost(11),revenue(6),duration(5),quality(0),flexibility(4),in([[3,9],[22,12],[39,13],[44,7]]),out([[16,3],
[36,7]]),mutual_exclusions([])).
task(id(306),cost(10),revenue(8),duration(12),quality(4),flexibility(2),in([[21,7],[23,9],[57,16]]),out([[20,10],
[62,5]]),mutual_exclusions([])).
task(id(307),cost(13),revenue(7),duration(8),quality(5),flexibility(4),in([[22,14],[46,9],
[69,1]]),out([[29,9]]),mutual_exclusions([])).
task(id(308),cost(12),revenue(7),duration(8),quality(3),flexibility(0),in([[12,3],[66,7]]),out([[6,13],
[22,18]]),mutual_exclusions([])).
task(id(309),cost(9),revenue(4),duration(9),quality(7),flexibility(1),in([[23,15],[39,2],
[55,7]]),out([[38,19]]),mutual_exclusions([])).
task(id(310),cost(10),revenue(5),duration(6),quality(1),flexibility(5),in([[1,4],[25,17],
[45,11]]),out([[58,2]]),mutual_exclusions([])).
task(id(311),cost(8),revenue(3),duration(12),quality(6),flexibility(5),in([[13,3],[27,11],[71,5]]),out([[29,14],[30,14],
[62,16]]),mutual_exclusions([])).
task(id(312),cost(15),revenue(12),duration(4),quality(3),flexibility(0),in([[15,6],[40,7],[45,12],[69,1]]),out([[39,12],
[43,13],[66,7]]),mutual_exclusions([])).
task(id(313),cost(8),revenue(3),duration(7),quality(6),flexibility(6),in([[20,14],
[52,10]]),out([[18,8]]),mutual_exclusions([])).
task(id(314),cost(17),revenue(11),duration(4),quality(1),flexibility(0),in([[52,1],[55,17],[57,18],[60,9],
[65,8]]),out([[1,14],[20,7]]),mutual_exclusions([])).
task(id(315),cost(15),revenue(12),duration(12),quality(3),flexibility(1),in([[23,19],[35,17],[38,18],
[59,11]]),out([[41,6],[63,4]]),mutual_exclusions([])).
task(id(316),cost(15),revenue(12),duration(9),quality(0),flexibility(7),in([[9,17],[48,12],[62,9]]),out([[35,15],[64,5],
[70,19]]),mutual_exclusions([])).
task(id(317),cost(13),revenue(10),duration(5),quality(2),flexibility(1),in([[13,5],[23,17],[30,10],[50,11],
[52,19]]),out([[16,5],[58,15]]),mutual_exclusions([])).
task(id(318),cost(11),revenue(9),duration(9),quality(6),flexibility(6),in([[10,18],[54,15],[63,3],[72,17]]),out([[14,14],
[46,9]]),mutual_exclusions([])).
```