FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Formal methods for reconfigurable assembly systems

Tiago Oliveira Ribeiro

Thesis submitted under the course of:
Mestrado integrado em engenharia electrotécnica e de computadores
Major in Automation

Advisor: Eng. Gil Gonçalves

February 2010

Resumo

As indústrias modernas têm uma contínua necessidade de satisfazer os seus mercados a melhores custos para se manterem competitivos. Este simples facto leva-as a desenvolver novas metodologias para a gestão e criação de novas linhas produtivas. Neste sentido, o projecto *XPRESS* pretende criar uma referência ao nível de plataformas inteligentes. O conceito subjacente ao projecto *XPRESS* são os seus agentes inteligentes hierquizados, os *Expertons*. Estes *Expertons* detêm o conhecimento necessário para realizar as tarefas que podem realizar e apenas precisam de ser instruídos sobre "o que fazer" e "quando o fazer".

Um outro conceito subjacente, está relacionado com a flexibilização das linhas de produção. Deste modo, a hierarquia de controlo foi pensada a maximizar a flexibilização dos equipamentos disponíveis no chão de fábrica. Um dos problemas relacionados com o controlo, prende-se com a forma de obter a escolha mais eficiente para a produção de um novo produto (ou variante).

Este trabalho tem como objectivo desenvolver um conjunto de métodos de análise e auxílio à escolha de *Expertons* (equipamentos) para novas linhas produtivas tendo como requisitos um novo producto.

Durante este trabalho foi desenvolvida uma biblioteca de funções de optimização de problemas combinacionais aplicado ao universo das indústrias baseadas em *Expertons*. A biblioteca é composta por três métodos, cada um correspondente a uma fase diferente do processo de optimização. O método *ReducedMatrix* baseia-se na medida da eficiência cruzada de cada um dos *Expertons* candidatos em conjunto com um processo de decisão multi-criterio. O método *ConfigurationGenerator*, por seu lado, é baseado numa estrutura em arvore com algoritmos de criação eficientes, o algoritmo *A Star*. Apesar do método *ConfigurationRanking* lidar com um conjunto de dados diferentes do método *ReducedMatrix*, o conceito matemático é analogo. Como tal, foi usada a mesma abordagem.

Este trabalho mostra, também, uma análise comparativa do impacto das características do problema sobre a implementação realizada, tal como o efeito da variação nos parametros de controlo dos algoritmos sobre a qualidade e rapidez de obtenção dos resultados.

Os resultados publicados nesta dissertação mostram que a estratégia de optimização proposta produz resultados viáveis sem penalização no tempo de execução do programa. Os resultados também mostram que, para problemas com menos de duzentos *Expertons* distribuídos por menos de dez tarefas, o efeito preemptivo do algoritmo *A Star* produz um impacto mais negativo na qualidade dos resultados que obtém do que a poupança temporal que fornece. Para problemas de tamanho semelhante ou inferior é, então, preferivel usar a versão não preemptiva analoga, o algoritmo *Best First*. Para problemas com um maior número de variáveis, a limitação ao nível da memória disponível no computador irá, definitivamente restringuir a utilização do algoritmo *Best First*. Para estes casos, não haverá outra opção senão usar o algoritmo alternativo, *A Star*, e aceitar os maiores riscos de não atingir as soluções optimas.

Abstract

Modern Industries have a continuous need to satisfy their markets at better costs to keep them competitive. This simple fact takes them to develop new methodologies for the management and for the creation of new produtive lines. On this purpose, the *XPRESS* project has the intention to create a standard on intelligent platforms. The concept behind the *XPRESS* project are hierquized intelligent agents called *Expertons*. These *Expertons* contain all the necessary knowledge to perform the tasks which they are able to do, and they only need to be instructed on "What to do" and "When to do it".

Another concept at the project is related with the flexiblization of the production lines. This way, the management hierarchy was thought to maximize the flexibility of the equipments available at the factory shop floor. One of the problems related with the management is what method to use so that the most efficient choice of *Expertons* for the production of a new product (or variant) is made.

This work has the objective to develop a set of methods for the analysis and assistance of the decision process regarding new production lines based on *Expertons* (equipments) towards the requirements of a new product.

During this work was developed a library of optimization functions for combinatorial problems applied to the universe of *Expertons* based industies. The library is composed by three methods, each corresponding to a different stage of the optimization process. The method *ReducedMatrix* is based on the cross-efficiency measure of each *Expertons* candidates together with a multi-criteria decision process. The method *ConfigurationGenerator*, on the other hand, is based on a tree data structure with an efficient creation algorithms, the *A Star* algorithm. Although the method *ConfigurationRanking* deals with a set of data structures different from the method *ReducedMatrix*, the mathematical concept beneath is the same. Therefore, it was used the same approach.

This work also makes a comparative analysis of the impact of the problem characteristics into the produced implementation. It also analysis the effect of the variation of the algorithm's control parameters over the results' quality and processor run time.

The results published on this thesis shows that the proposed optimization procedure produces viable results without penalizing the run time of the software. The results also show that, for problems with less than two hundred *Expertons* distributed along less than ten tasks, the preemptive effect of the *A Star* algorithm produces a higher negative impact on the quality of the results than on the time saving it provides. For problems with sizes similar to this or smaller, it is then prefed to use the analogue non preemptive version, the *Best First* algorithm. For problems with a higher number of variables, the limitation of the computers memory will, for sure, represent a handicap for the Best First algorithm. This case, there is no other option than using the A Star alternative and accept the risk of not achiving the optimal solutions.

Acknowledgements

I would like to take this opportunity to thank all the support, knowledge and recommendations I received during this study.

Professor Gil Gonçalves and his guidance and support helped to the success of this work.

The group which works for *XPRESS* on FEUP : Paulo Dias, Bruno Terra and Fernando Almeida, for their insights of the *XPRESS* project.

And, finally, to Critical Manufacturing on their support during the integration phase.

The Author

Contents

List of Figures

List of Tables

Abbreviations

AGV	Autonomous Guided Vehicle
BF	Breadth first
BoM	Bill of materials
BoP	Bill of processes
CE	Configuration Experton
CCR	Charnes, Cooper, and Rhodes model
DF	Depth first
DLL	Dynamic-link library
DMU	Decision Making Unit
eBOP	electronic Bill of Process
ENF	Expertonic Network Factory
ERP	Enterprise Resource Planning
ESD	Experton Self Description
FEUP	Faculdade Engenharia Universidade Porto
HMI	Human Machine Interface
GLPK	GNU Linear Programming Kit
GUI	Graphical User Interface
KPI	Key Performance Indicator
NPI	New product implementation
PCS	Production Configuration System
PE	Production Experton
PES	Production Execution System
PQS	Production Quality System
PSS	Production Simulation System
QRD	Quality Report Document
QM	Quality Manager
SMED	Single Minute Exchange Die
TCO	Total cost of ownership
TDD	Task Description Document
TE	Transport Experton
UML	Unified Modeling Language
VCN	Value Chain Networks
WES	Workflow Execution System
WF	Workflow
WFM	Workflow Manager
WIP	Work in progress
WWW	*World Wide Web*
XML	Extensible Markup Language
5S	Seiri, Seiton, Seiso, Seiketsu, Shitsuke

Chapter 1

Introduction

The industrial manufacturing is characterized by continuous improvements and by innovation cycles induced by a global competition striving for the same goals. A group of industries, in conjunction with information and communication technology providers, and academic and research institutions, constituted a consortium to develop an european integrated project called *XPRESS* - Flexible production experts for reconfigurable assembly technology.

The *XPRESS* project has the goal of establishing a breakthrough for the factory of the future with a new flexible production concept based on specialised intelligent process units, called Expertons, integrating a complete process chain. The project concept is focused on the automotive, aeronautics and electrical industries, due to the members of the consortium, but its concept can be transfered to nearly all the production processes.

One of the problems during the set up of a process line is to choose the best combination of the equipments from the ones available at the plant *or even on the market*.

The goal of this thesis is to provide, the *XPRESS* project, a set of methods to assist the planning process, in which the best combination of equipments (Expertons), available at the plant or even on the market, is achieved to produce a certain product and its variants.

1.1 Research Objectives

The main goal of this work is to provide a set of methods to assist the engineers' decision regarding the choice of *Expertons* to use for a specific job description.

Another aim of this research is to provide a comparative study of the several alternatives and their main characteristics.

1.2 Scope and Limitation of the Research

The research presented here focuses on defining a set of methods capable of reducing the space solution of combinatorial problems, and requiring the minimal impact on the goals to achieve. It is also the aim of this research to be a tool for the planning activities of new process lines.

An introduction to the mathematical concept underlying this problem is made during this document. Further analysis can be made through the external references.

The literature used during this research is mostly related with similar investigations, however, only a small description, with the revelant information, is made.

The output of this work is meant to be integrated into the european investigation project - *XPRESS*. The process of gathering the information from the several sources, such as, expertons, knowledge databases, simulation tools is out of the scope of this work. The graphical user interface, usually available on the tools of this kind and which will manage the workflow of the planning process, is also out of the scope of this work.

The *XPRESS* project, which is constituted by consortium of a group of industries, in conjunction with information and communication technology providers, and academic and research institutions is developing a functional sample, which will provide real scenarios and equipments. Due to the differences on the timeframes between the *XPRESS* project and this thesis, the results of this activity are not available, thus they cannot be presented yet.

1.3 Document Overview

This thesis describes the development of a package of methods for the selection procedure of a set of expertons towards a task goal. The several stages of development will be discussed through the entire document.

This dissertation is composed by seven chapters. On Chapter 2, the context of the research is analysed and related work is reviewed and discussed. Chapter 3 analyzes the system requirements and the design stage of the algorithms. The development stage of the library package and the complexity analysis of its algorithms is detailed on Chapter 4. On Chapter 5 are presented the functional validation method, as well as, the analysis of the algorithms' performance and characterization of their main variables. On Chapter 6, aspects related with the project integration are shown. Final review and conclusions of this thesis are presented on the last chapter.

Chapter 2

Analysis of Relevant Literature and Related Projects

This chapter describes the motivation for the development of new methodologies for the industries. It also describes a new industrial concept. Finally, there is an introduction to the operation research's assignment problem and a brief review of other examples, in which this mathematical concept is also applied.

2.1 Industries State of the Art

Industries greatest mote is profitability. But profitability cannot be given for granted along time. Therefore, industries strive for being more efficient, productive and cost effective to be able to resist to a tough competition, even during harsh world economics downturns.

Today's industries set up their production lines either on a process based on configuration or on a product variant. This means that for each new product variant, the engineer of each area has to develop a new recipe. The new recipies are mostly copied from similar previous versions and only a few parameters are changed.

This approach is reliable and does not cause disruptions on the quality performance of the lines. On the other hand, it is neither optimal, nor the most cost and time efficient.

Another problem today's industries face is to adapt to market expectations on new features and the possibility of costumization, which leads to a wider product portfolio and to smaller lot sizes, into the production process. On the shop-floor increases the complexity of managing the production, as well as, the cost due to the reduction of the economy of scale and the equipment downtimes due to set ups.

Flexibility at equipment or process side is many times considered to be "nice to have" but whenever huge costs are involved, it is many times neglected due to the pressure from the market competition to produce the same product, with the same quality at a cheaper price to the consumer. And many industries simply introduce a new variant when the expected number of sales is enough to pay the addition of a new process line, or when there is a vacant process line from a deprecated product.

As lines are set up mainly concerning cost versus flexibility, the introduction of a new product, or even a product variant, causes a disruption of the production flow. Whenever a variant requires an equipment to change setups, including changing or adjusting accessories, there is a list of performance indicators which may deteriorate. A tipical list of problems would be:

- An increase on downtimes.
- An increase on human intervention leading to higher probability of human errors.
- An increase of time losses.
- An increase of the scraped material or of the rework to defectuous parts.
- A decrease of quality due to worn parts, which is not so easly noticed due to constant adjustment for different products.

To attack most problems, engineers employ methodologies like Lean Manufacturing, 6 Sigma, or even combined methodologies like Lean 6 Sigma. Those methodologies try to identify points of improvement on the processes and then propose solutions. To sum up, they produce continuous improvements on the product quality and, at the same time, they stablize the process and produce less waste.

The SMED (Single Minute Exchange Die) methodology [3] focuses on the activities done during the set up of machines to minimize the down time. It is meant to identify the tasks that can be performed without having to stop production (external tasks), and even changing the procedure so that an internal task can become external. A famous example of what SMED means is the way formula 1 cars are refuelled and the way tires are changed during races compared to everyday cars - a typical operation of a dozen minutes can be performed under ten seconds.

The SMED methodology is usually applied together with the 5S methodology [4]. The 5S focuses on the workplace environment, organization and safety, specially when people are involved.

Another consequence of these design methods is the usage of batch processing or the usage of lots with large quantities. Although not punishing the up time of the particular process or equipment, it generates large queues of material waiting for processing (WIP) and degradates the product cycle time.

All these methodologies are reaching the limit of its capabilities, and industries are searching for the next improvement leap.

A chronological representation of these improvement leaps is presented on the figure 2.1

The *XPRESS* project is one of the future possibilities. With the flexibility on its core, the hierarchy of the industrial systems will change once more. What was a rigid structure in the past, is now reconfigurable system of higher degree. Figure 2.2 shows some of these evolutions over time.

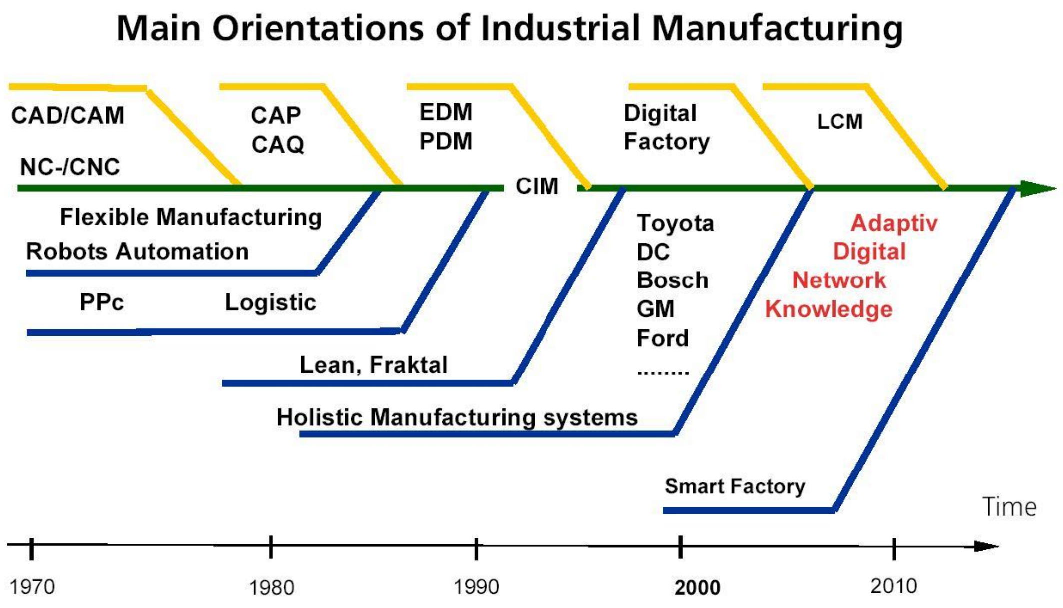


Figure 2.1: Industrial manufacturing orientations over time [1]

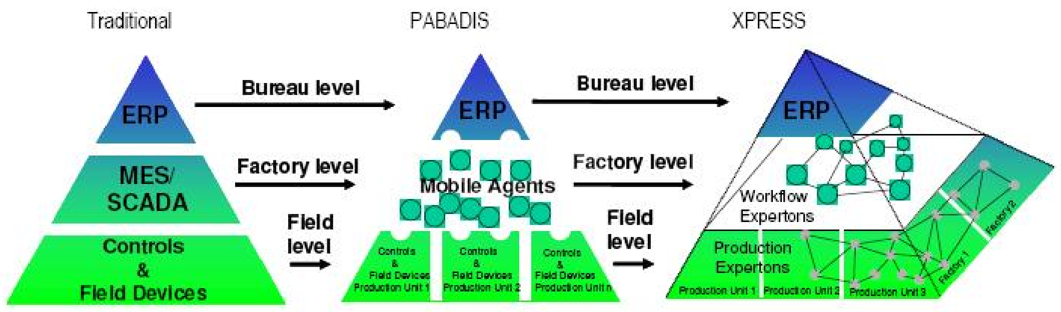


Figure 2.2: Transition from rigid to reconfigurable systems and new *XPRESS* dimension [1]

2.2 The *XPRESS* project

The industries needs for inovation is a continuous and never ending task. One of many present attempts to develop new tools for the industries is the *XPRESS* project.

The goal of *XPRESS* (described at its Description of Work [1]) is to establish a breakthrough for the factory of the future with a new flexible production concept based on the generic idea of "specialised intelligent process units", *Expertons*, integrated in cross-sectoral learning networks for a customised production. This knowledge-based concept integrates the complete process chain: production configuration, multi-variant production line and 100% quality monitoring. This concept is demonstrated both in the automotive, aeronautics and electrical industries but it can be transferred to nearly all production processes.

XPRESS meets the challenge to integrate intelligence and flexibility at the "highest" level of the production control system as well as at the "lowest" level of the singular machine.

The radical innovations of the "Expertonic networked factory" are knowledge, responsibility segregation and trans-sectoral process learning in specialist knowledge networks.

The concept is built on coordinated teams of specialised autonomous objects *Expertons*, each knowing how to do a certain process optimally. They have the intelligence to choose the best known production parameters for a given task.

Assembly units composed of Expertons can flexibly perform several types of complex tasks, whereas today this is limited to a few pre-defined tasks.

Communicating machines will be developed using self-organising and self-learning algorithms, data mining and common interfaces.

By sharing the specific knowledge of each Experton in a network, other Expertons are not only able to learn from each other in one production line, but also between different lines and in different production units.

This architecture allows continuous process improvement.

XPRESS will be able to anticipate and to respond to rapidly changing consumer needs, producing high quality products in adequated quantities while reducing costs.

The *XPRESS* project will address the following strategic objectives:

1. To establish a breakthrough for the factory of the future with a new flexible assembly and manufacturing concept based on the generic idea of "specialised intelligent process units", *Expertons*, integrated in cross-sectoral learning networks for a customised production and a flexible system organisation. This knowledge-based concept integrates the complete process chain from the production planning to the assembly, and finally to the quality assurance of the produced/assembled products and to the reusability.

2. To meet the challenge of future intelligent manufacturing concepts, to integrate intelligence and flexibility not only at the "highest" level of the production control system but also at the shop-floor level.

3. To demonstrate the feasibility in 3 representative applications (automotive, aeronautics and electrical industries).

XPRESS also presents an "Objective-driven approach". This new concept will be developed and demonstrated by a strong industry-lead partnership in order to meet the still remaining industrial needs with regard to:

1. Production Configuration and Simulation: *XPRESS* will decrease the ramp-up time for assembly lines by at least 50% and increase the reusability of assembly components and optimise the entire assembly process

2. Experton guided Production flow for the assembly and manufacturing of different types and variable volumes of products on a single flexible line and achievement of a high level of reusability.

3. Expertonic Machines and Human Integration:

   (a) reducing the effort needed for setting up a single process,

   (b) providing most efficient and reliable inline quality assurance systems for the processes,

   (c) reacting intelligently in case of disturbances

   (d) providing a factory-wide process monitoring system, and

   (e) allow the reuse of disassembled components.

2.2.1 What is an Experton?

"An Experton is a self-containend entity, which is encapsulating expertise and functionality and interacts with its environment by exchanging standardized, synchronous messages." [1]

An Experton is an agent which decides how to reach its given goals best, but not when to do it. Therefore, an Experton is an equipment which has all the knowledge to perform a certain task, and that only needs an instruction on what and when to do, but not the how to do it. Each experton is also able to respond if it is capable or not to perform the operation on that particular manufacturing conditions.

The task execution is triggered from the outside as defined by another Experton category, named "workflow manager" (WfM), overlooking the factory level with dedicated knowledge expertise. Expertons can be hierarchized into three categories:

- *Configuration Expertons:* Responsible for finding an optimum production configuration and for the creation of a Workflow manager template which can then be called to produce the product variant.

- *Workflow manager:* Controls the production flow of an item according to the manufacturing execution template

- *Production Expertons:* Responsible for executing basic manufactoring tasks and for coordinating groups of production expertons.

Expertons can communicate with others according to this hierarchy through three channels/documents:

- The *Experton Self Description (ESD)*, the means on which the experton can describe his capabilities and caracteristics.
- The *Task Description Document (TDD)*, the communication channel in which one experton requests a task to be performed by an another experton. It is here that particular conditions are specified.
- The *Quality Result Document (QRD)*, the communication channel where the reply of a TDD is given; it represents the quality results of the task that was performed.

These documents are XML files whose structure is agile enough to be universal. The content of this XML files are then adapted to the circustances and to their destination.

2.2.2 PCS/PSS/PES

The configuration experton objective is to create a workflow(WF) manager instance, able to produce the part as described on the production procedure as well as the necessary handling of materials between stations and process steps, and constrained by the planning and simulation rules. This WF instance must correspond to the best combination of expertons as defined by the cost function.

The PSS is a combination of four functions interconnected as the figure 2.3.

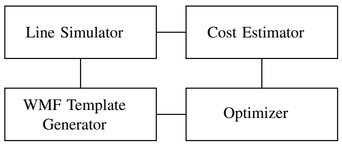


Figure 2.3: PSS block diagram

The production configuration system is finding the best fitted assembly arrangement for a given product under defined boundary conditions. It is composed by *Expertons*, supplying the functionality of simulation and cost estimation, and by a core component, distributing the information streams of the configuration system and producing workflow manager agents controlling the production flow of product items, the configuration Experton (CE).

The production configuration system (PCS) is the agent which defines the optimum configuration of an assembly arrangement during the planning phase, based on a definition of the product variants and processes, production goals and Production Experton candidates. After ramp-up the PCS will produce workflow managers via agent factory, which are controlling the assembly arrangement downstream in order to produce one specific product item each. Disturbances in the assembly configuration (like broken Production Experton) will be compensated by the PCS by finding a new (sub-optimal) configuration using the same process as during the planning phase, but with restricted search space. The abilities of the CE are:

- Production of an "electronic Bill of Product" (eBOP), a complete description of the whole assembly process including the requirements for all steps
- Instantiation of Workflow Manager (WfM) agents with an eBOP during a new product implementation (NPI) process,
- Instantiation of permanently adjusted WfMs to account for changes in the production conditions.

In order to fulfil its purposes, the configuration Experton has to gather the relevant data for the assembly of a given product and find the best production method sequences (e.g. forming, joining, etc.) by the use of knowledge bases, simulation and cost estimation. From the result of the optimization, a WfM for the optimum production process can be instantiated.

The PCS will have the following components:

- Simulator: Simulation of workflows with different Production Expertons (PE) and work unit configurations. Gathering of data needed for simulation. During simulation, virtual PEs are simulating their work in real time, using and delivering the same information via the same interfaces as in real operation. The simulation Experton will inherit simulation algorithms and will be able to correspond with external simulation tools. An upstream analysis of simulation data will be incorporated to restrict the amount of data which has to be transferred.
- Cost estimator gives the weighted cost of the simulated production and difference from the production goals as an input to an optimiser. The cost estimator Experton will work with state-of-the-art algorithms and methods. Data from the simulation and the Production Expertons will feed the cost estimation process. If some data is not available, assumptions can be made based on the knowledge base.
- Optimiser finds the optimum process, finally described by the eBOP.
- Configuration Experton as the central information hub for all components allowing plug-and-play exchange. Production of workflow managers.

The PSS will be the top level representation of the line. With the PSS not only a new manufacturing line can be simulated in advance but also existing manufacturing lines can be simulated to deduce results e.g. for a specific product mix envisaged in the future. The PSS will be responsible for reflecting the overall behaviour of the line. The key performance indicators (KPI) of the model will be e.g. cycle time, influence of failures, influence of product mix, work in progress (WIP) in the line, etc. For detailed analysis of individual elements of the line (such as cells) the PSS will be able to activate simulations on other levels. The results of these simulations will be analysed and processed to serve as input and/or reference for the line simulation. The PSS will incorporate as much of a generic digital model as possible. However, as all available simulation tools have a proprietary data structure, some parts of the digital model have to be modelled in the simulation tool. Through the use of building blocks, this influence will be lessened.

From the digital model in the PCS, the PSS will derive all necessary input data. Therefore the input data will be always up-to-date.

The work will be carried out by defining an appropriate simulation tool which can demonstrate the feasibility of the approach. This tool will be wrapped into an Experton and fitted out with interfaces to the Expertonic system. The results from the line simulation will be analysed and stored inside the PCS knowledge base. For effective storing and enhancement with appropriate context, the results have to be processed and condensed, as much as possible, to avoid large amounts of data.

One output of the PSS will also be the template for the Workflow Manager Experton. This will be stored in the PCS knowledge base, to be available for the execution of the manufacturing tasks (by the PES (Production Execution system)).

Two major scenarios will be covered by the Configuration Experton: the re-configuration of an existing line and the set-up of a new manufacturing line. For the re-configuration scenario the Configuration Experton will use all available input from the Expertonic system.

Information from existing Expertons like the Super-Experton will be accessed. Furthermore the inferior Expertons can provide all data from their operational history such as cycle time, breakdown behaviour, accuracy, control programs, etc. With this data, the planner will be in position to carry out the planning tasks much more efficiently than today.

The access to knowledge bases, via the interfaces, will also be provided. Through the knowledge bases the planners do not have to circuitously search for the needed information. The same will apply for the configuration of new (non-existing) manufacturing lines. Working in a very similar way to the re-configuration scenario, this one has to deal with a lot more uncertainty regarding all entities and facilities that will be used. Therefore, the collaboration among the planning team will be vigorously supported by the PCS, which will provide all knowledge through knowledge bases for the planners, starting from layout and available equipment to the finding of feasible process parameters.

An important role for the Configuration Experton will also be the incorporation and coordination of external tools provided by other suppliers. These tools, in the field of work of CAD, CAM and likewise, are used today and shall not be replaced. Instead they will be incorporated and fitted, where possible, with interfaces to the PCS.

The Cost estimator Experton is mainly responsible for the cost estimation of a specific set of line configuration. These costs are influenced by two main aspects: One of them, the total cost of ownership (TCO), is monetary, and the other is related to the key performance indicators (KPI), like cycle time, work in process (WIP), etc. For the estimation of the costs, the representation of a digital model of the factory and all its elements is very important. Not only new equipment, but also already existing one, will be, due to the Expertonic self-description capability, able to provide their actual accounting worth. An equivalent model has to be generated for costs which are usually not represented in the Expertonic system. These costs comprise facilities, part of the workforce and other running expenses. The Cost estimator Experton will define the costs after simulation and will therewith provide the necessary input data for the optimization algorithm.

The Production Execution System (PES) is the operational part of the PCS. In contrast to the Production Simulation System, no further simulation will be done. The main task of the PES is the instantiation of the Workflow-Managers based on the Workflow-Manager templates that were stored in the commonly used PCS knowledge base.

Therefore, it is necessary that the PES has got an additional interface to access information from external systems like the ERP system. The order in which specific data will be provided by those systems consists of e.g. the amount of pieces to be produced as well as some detailed data that were of no interest during the planning phase, like the colour of a part.

After receiving an order, the PES will look into the PCS knowledge base for a WfM-template that matches the requirements described in the order TDD. For each TDD that will be executed in the PES, there is already a fitting template, which was stored there by the PSS. Those templates will then be enriched by further information from the ERP system.

Having all necessary data as an output, the PES will instantiate one Workflow-Manager for each unit of parts that has to be produced and will handle it to the Workflow Execution System (WES).

2.2.3 WES

The workflow manager Experton consists of two basic components: the first component is the workflow modelling component, which will enable the definition of the processes and activities, analyse and simulate them, and assign them to other Expertons. The second component is the workflow execution component. The run-time system will consist of an execution interface which will stimulate the required production and other Expertons to carry out the work.

The workflow execution system is subdivided into two parts: the WFM and the QM.

The WFM is responsible for issuing the TDD's for the production expertons and its proper scheduling.

The QM is responsible for storing the information of the QRD gathered as responses of the TDD's sent.

2.2.4 Production Expertons

The production expertons are the equipments available at the shop-floor which produce work. They are classified according to the main task they can perform:

- Transport Experton: Responsible for the transport of materials and finished goods between stations and between storage points. Within XPRESS, two kinds of Transport Expertons will be investigated: Transport Expertons based on belt conveyors and Transport Experton based on Automated Guided Vehicles (AGV).

  - AGV based Transport Expertons: The special capabilities of AGV based Transport Expertons are in the planning of routes on which AGVs transport products. In case of redundant production equipment or very flexible assembly sequences, this also includes an intelligent prediction: to which cell the product should be transported next. Thus, those Expertons do not have a static local position connecting dedicated production Expertons, but are moving according to supply requirements. This also means, that they are not related to one production cell like the other Production Expertons. Transport Expertons have the capability to plan routes and paths by themselves.
  - Conveyor belt based Transport Experton: In contrast to AGVs based Transport Expertons, conveyor belt based transport Expertons do have a fixed position in the production environment. However, those Expertons are also supposed to take a product from one production cell to another. For that, they also do not logically belong to one production cell and thus, they are not part of a Super-Experton controlling them. This case, in which the transport is an inherent, internal property of a Super-Experton, is not considered here. The special requirement on conveyor belt Transport Expertons is the capability of carrying more than one product at the same time. This means that the position of each product carried by the conveyor belt at the same time must be tracked by a tracking system, in order to deliver process and quality data to the Quality Manager.

  Transport Expertons may also form a hierarchy with separated responsibilities for the individual execution of a single transport task or the responsibility for the dispatchment of transport processes.

- Handling Experton: Manipulates the materials within the cell.
- Process Expertons: such as Milling, Drilling, Welding, Riveting, Bending expertons.

2.2.5 The Super-Expertons

The Super-Expertons combine information from their adjacent Expertons like handling or welding Expertons.

The Super-Experton will also be able to provide results based on internal simulation.

2.2.6 Knowledge Base

Included on the Expertonic Network Factory, there is a database containing information gathered along time. Among others, it contains information of all the simulations that have been done: all the configurations that are and that were on production.

Each individual experton also has its private knowledge base to store information of his process. This data is used by the experton to develop its best methods for performing a given task.

2.3 Operational Research

The main purpose of this work is to find optimal processes and, in fact, Operational Research tools exist for this purpose. Therefore, some of them will be analised on this section.

2.3.1 The assignment problem

The assignment problem is a special type of linear programming problem where resources are being assigned to perform tasks [5].

To fit the definition of an assignment problem, this kind of application needs to be formulated in a way that it satisfies the following assumptions:

1. The number of assignees and the number of tasks is the same.
2. Each assignee is to be assigned to exactly one task.
3. Each task is to be performed by exactly one assignee.
4. There is a cost c_{ij} asssociated to the assignee i performing the task j.

Despite this, most of the problems do not fit these requirements. As an example, sometimes the number of assignees and the number of tasks is different. On situations like this, the assignment problem is called unbalanced. To make it balanced, it is frequent the usage of dummy assignees or dummy tasks.

Being X the assigment matrix, where:

$$x_{i,j} = \begin{cases} 0 & \text{if assignee i performs task j} \\ 1 & \text{if not} \end{cases}$$

And C the cost matrix, where:

$$c_{i,j} = \text{the cost of the assignee i performing the task j}$$

The assigment problem model is on equation 2.1.

$$Minimize\ Z = \sum_{i=1}^{n}\sum_{i=1}^{n} c_{i,j}x_{i,j} \tag{2.1}$$

Subject to the restrictions on equations 2.2.

$$\begin{aligned} \sum_{j=1}^{n} x_{i,j} &= 1 \quad \text{for } i = 1,2,\ldots,\text{n}, \\ \sum_{i=1}^{n} x_{i,j} &= 1 \quad \text{for } i = 1,2,\ldots,\text{n}, \\ x_{i,j} &\geq 0 \qquad \text{for all } i \text{ and } j \end{aligned} \tag{2.2}$$

The graph representation of the problem is described on figure 2.4

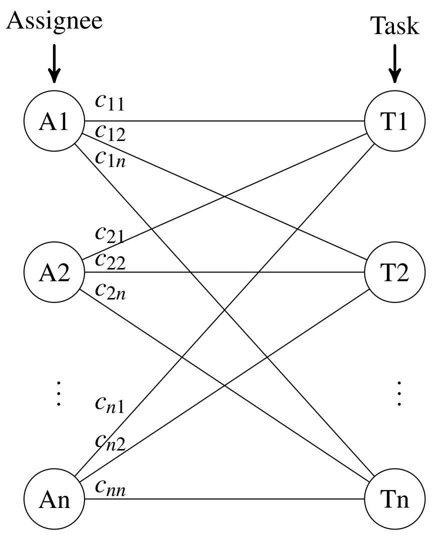


Figure 2.4: Graph representation for the assignment problem

When an assignment problem respects all these conditions, there is a simple algorithm to efficiently evaluate the solution. This algorithm is known as the Hungarian Method [6].

Step 1 Find the minimum element in each row of the $m \times m$ cost matrix. Construct a new matrix by subtracting from each cost the minimum cost in its row. For this new matrix, find the minimum cost in each column. Construct a new matrix (called the reduced cost matrix) by subtracting from each cost the minimum cost in its column.

Step 2 Draw the minimum number of lines (horizontal and/or vertical) that are needed to cover all the zeros in the reduced cost matrix. If m lines are required, an optimal solution is available among the covered zeros in the matrix. If fewer than m lines are needed, proceed to Step 3.

Step 3 Find the smallest nonzero element (call its value k) in the reduced cost matrix that is uncovered by the lines drawn in Step 2. Now subtract k from each uncovered element of the reduced cost matrix and add k to each element that is covered by two lines. Return to Step 2.

2.3.2 Goal programming

Goal programming is an extension of linear programming used on the context of multiattribute decision making [6].

On a decision with n attributes and where an additive linear cost function can be defined as $c(x_1, x_2, \dots, x_n)$, being x_i the value of the alternative for the attribute i. If A represents the decision maker's set of possible decisions, then the best alternative, a^*, satisfies the condition 2.3

$$min_{a \in A}\ c(x_1(a), x_2(a), \dots, x_n(a)) = c(x_1^*(a), x_2^*(a), \dots, x_n^*(a)) \tag{2.3}$$

A decision maker's preference can only be represented by an additive cost function if the set of attributes 1, 2, ..., n is mutually preferencial independent.

$$c(x_1(a),x_2(a),\ldots,x_n(a)) = \sum_{i=1}^{i=n} c_i(x_i) \tag{2.4}$$

A solution A to a multiple-objective problem is Pareto optimal if no other feasible solution is at least as good as A with respect to every objective and strictly better than A with respect to, at least, one objective.

2.3.3 Cross-reference comparision

A specific data development analysis model referred to as Charnes, Cooper and Rhodes (CCR) [7] model is a fractional programming technique that evaluates the relative efficiency of homogeneous decision making units (DMU). The general efficiency measure can be best summarised by the Equation 2.5

$$E_{ks} = \frac{\sum_y O_{sy} v_{ky}}{\sum_x I_{sx} u_{kx}} \tag{2.5}$$

O_{sy} are the output measures of the DMU s

v_{ky} are the output weights of the "target" DMU k

I_{sx} are the input measures of the DMU s

u_{kx} are the input weights of the "target" DMU k

E_{ks} is the cross-effiency of DMU s, using the weights of "target" DMU k

An optimization using the cross-reference comparison is obtained E^*_{kk} by maximizing the Equation 2.6 with the conditions on Equations 2.7.

$$E_{kk} = \frac{\sum_y O_{ky} v_{ky}}{\sum_x I_{kx} u_{kx}} \tag{2.6}$$

$$E_{ks} = \frac{\sum_y O_{sy} v_{ky}}{\sum_x I_{sx} u_{kx}} \leq 1 \forall s \tag{2.7}$$

$$v_{ky} \geq 0$$

$$u_{kx} \geq 0$$

If E^*_{kk} is equal to 1 then there is no other DMU which is better than DMU k for it optimal weights.

In spite of this, this optimization is not solved through the usage of linear programming methods. There is a simplification that can be done, which is to choose for u_{kx} such values that $\sum_x I_{sx} u_{kx} = 1$.

Solving this optimization to all the DMUs, then it is possible to select the ones which are not optimal ($E^*_{kk} < 1$) and remove them from the space solution.

The cross reference comparison may lead to pareto optimal solutions but it is not a sufficient condition, because it eliminates solutions which are strictly better than them with respect to, at least, one objective but it cannot guarantee that it eliminates a solution *A* when another feasible solution is, at least, as good as *A* with respect to some objectives and strictly better than *A* with respect to, at least, one objective.

2.4 Team formation approach

The conceptual idea of assigning an equipment to a job specification could also be seen as selecting a group of elements for a team to achieve a goal.

Therefore, projects related to team formation were analysed regarding the usage of algorithms.

The work by Talluri *et al.* [2] defines a two phase procedure to create strategic interorganizational networks. On its first phase, the method used to evaluate candidates is using a cross-reference measure, described in the subsection 2.3.3. On its second phase, a integer linear programming is used to determine the best combination. This study also propuses the creation of Value Chain Networks (VCN) based on the diagram on figure 2.5. Although the context of the work is focused on interorganizational networks, industrial manufactures are also composed of VCN. Furthermore, the analysis, requirements and informational flow are analogues.

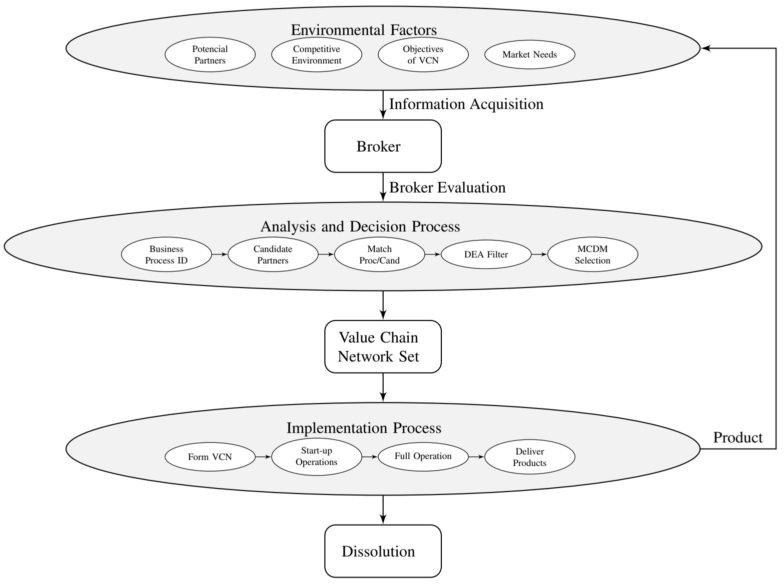


Figure 2.5: Value Chain Network design [2]

Goal programming is also used in other similar works for the team selection, such as Hajidimitriou and Georgiou [8]. However, it focuses on defining financial measures and defining a constant cost function, on which only variable parameters are changed on run-time and, not on evaluating, if goal programming was the best tool for the job.

In Zakarian and Kusiak [9] present a conceptual approach for selecting, among members of a team with different capabilities, the most adquate ones regarding the clients requirements and product characteristics. Through a tree phase approach, planning matrices and team members characterization matrices are used as inputs for an optimization problem, represented on figure 2.6, which then determines the optimal team composition.

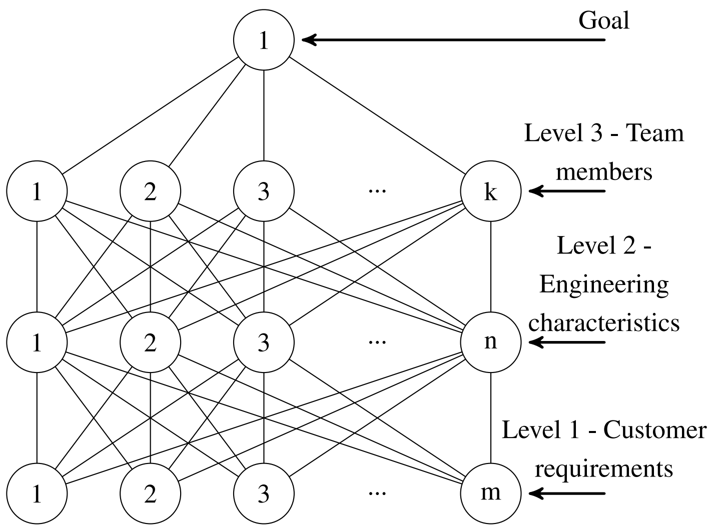

Goal
Level 3 - Team members
Level 2 - Engineering characteristics
Level 1 - Customer requirements
1
1
2
3
...
k
1
2
3
...
n
1
2
3
...
m

Figure 2.6: Hierarchical structure for the team selection problem

Other projects focus on other factors, like dynamically changing systems. On Gaston and Jadins [10], team formation is applied on a context of dinamically changing organizations. This means that the set of job descriptions, at a given time, is dynamically adapted and that the process of setting up a new job is based on gathering a set of agents capable of doing it. Then, they propose two possible strategies: structure based adaptation or performance based adaptation strategy.

Tseng *et al.* [11] apply a methodology for multi-functional project teams based on fuzzy logic, which is useful when there is no clear relation between the project characteristics and the client requirements.

Other methodologies are also used. For example, Hexin and Jian [12] define a methodology based on the risk assessment for the partner selection; its selection method is based on genetic algorithms.

The study by Tidhar *et al.* [13] describes a mechanism for team selection in a multi-agent system. This mechanism is decomposed into two main steps: being the selection of a team that will attempt to achieve the goal the first, and the selection of a joint plan of action the second. The designer has the ability to guide the process by typing the agents and teams and their specifications towards a given goal. This approach is not suitable for time critical domains.

2.5 Graph Grammars

Graph Grammar is a method for representing systems and their evolution along time, by using graphs and transformations over graphs [14].

A labeled graph is a triplet $G = (V, E, l)$ where V is the set of Vertices or Nodes, E is a set of pairs or edges from V, and l is a labeling function of V over an alphabet Σ.

Graph Grammar is composed by a set of rules Φ, which represent all the actions that might take place over the system and an initial scenario G_0. The system (G_0,Φ) defines a universe of scenarios G_i, which are obtained by the application of a rule Φ_n to the graph G_{i-1}.

Graph Grammar defines systems which are non-deterministic dynamical systems, due to the fact that, at any time, more than one rule might be simultaneously valid to the graph, but then only one can be applied. Moreover, as there is no priority or preference, the rules have no characteristic that defines which one should be applied first.

The properties of a system regarding convergence and stability are a consequence of the rule set Φ.

An example of a graph grammar could be used to describe a transportation network. On this example, there are three kinds of nodes: Stations (s), Vehicles (v) and Tracks (t). The connection between Stations and Tracks defines the transportation layout, and the connection between Vehicles and Stations or Tracks defines the current position of the vehicles on the layout. On figure 2.7 is the initial state of the system, G_0:

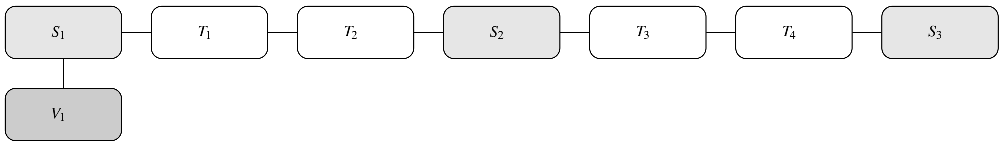


Figure 2.7: Graph Grammar: G_0 representation

In this case, the vehicle can move freely between stations and tracks, and among tracks. This is defined by the set of rules on figure 2.8

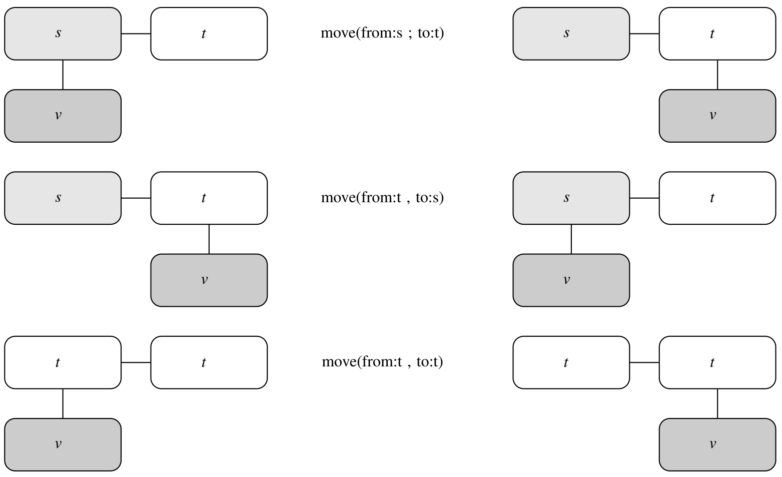


Figure 2.8: Graph Grammar: rules representation (1)

And, to finish the rules, the load and unload operations move material from and to the stations. The material, in this case, is not represented as a node (which was possible) but as a property of the nodes station and vehicle as described on figure 2.9

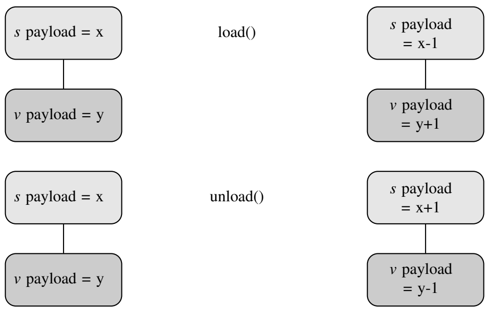


Figure 2.9: Graph Grammar: rules representation (2)

Have in mind that the representations from figure 2.11 until 2.20 are only one of the several possibilities for the initial system. Another example would start by moving the vehicle to station S_2 and then perform the first load.

Another example of the usage of graph grammars on transport networks is present on the study Añez *et al.* [15]. This study defines a dual graph transformation, particularly useful for the representation of complex transport networks when there are numerous turn prohibitions or restrictions.

One of the potencials of graph grammars is its use to describe autonomous system. The study by Klavins *et al.* [16] defined a class of graph grammars that describe self assembling graph systems. It focuses on the properties of the rule set to produce reachable and stable graphs.

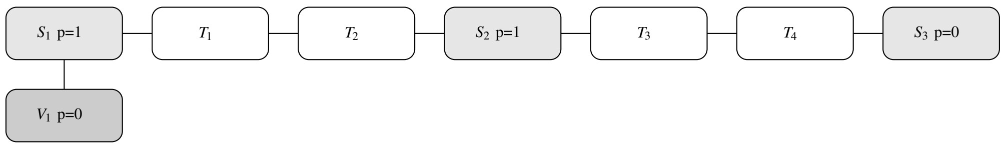


Figure 2.10: Graph Grammar: G_0 representation updated with payload

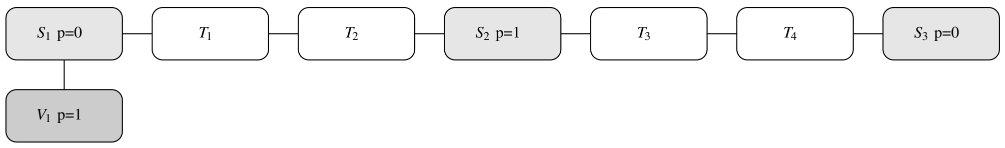


Figure 2.11: Graph Grammar: G_1 : load() $\rightarrow G_0$ representation

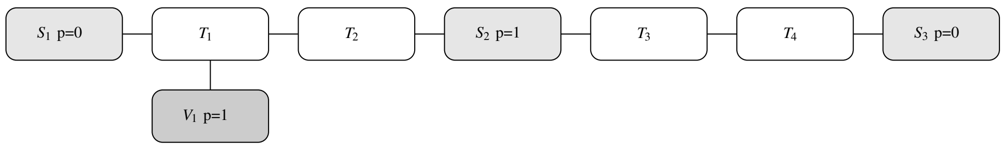


Figure 2.12: Graph Grammar: G_2 : move(s,t) $\rightarrow G_1$ representation

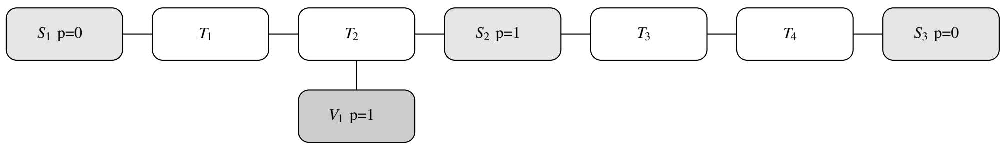


Figure 2.13: Graph Grammar: G_3 : move(t,t) $\rightarrow G_2$ representation

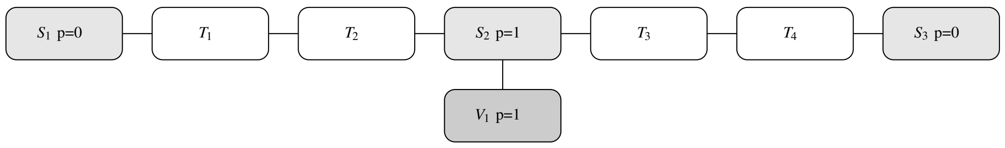


Figure 2.14: Graph Grammar: G_4 : move(t,s) $\rightarrow G_3$ representation

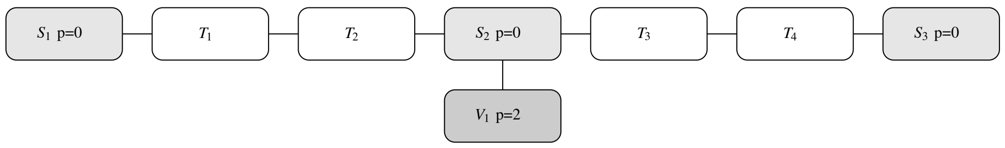


Figure 2.15: Graph Grammar: G_5 : load() $\rightarrow G_4$ representation

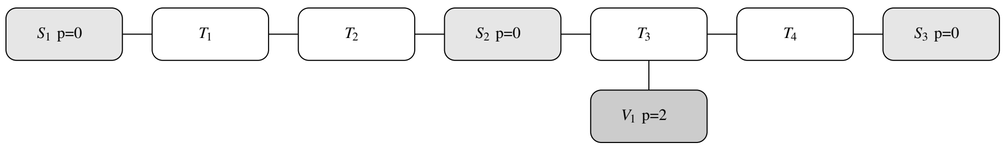


Figure 2.16: Graph Grammar: G_6 : move(s,t) $\rightarrow G_5$ representation

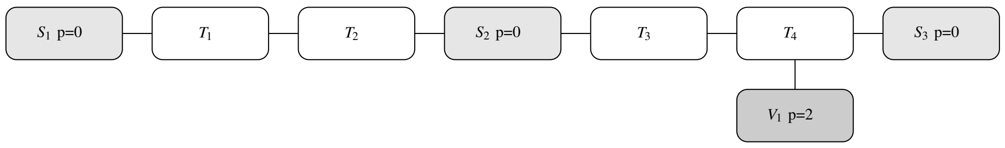


Figure 2.17: Graph Grammar: G_7 : move(t,t) $\rightarrow G_6$ representation

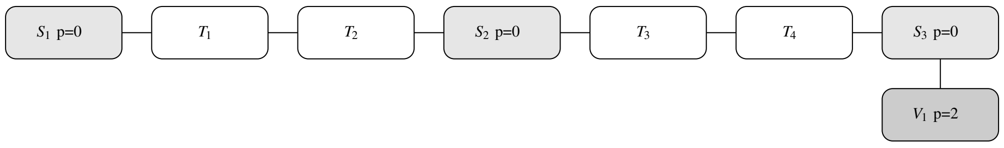


Figure 2.18: Graph Grammar: G_8 : move(t,s) $\rightarrow$ G_7 representation

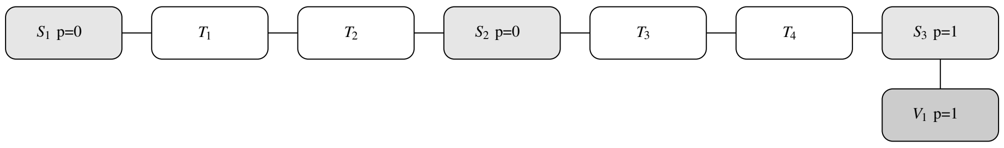


Figure 2.19: Graph Grammar: G_9 : unload() $\rightarrow$ G_8 representation

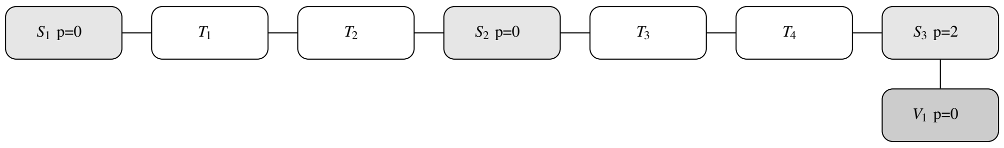


Figure 2.20: Graph Grammar: G_{10} : unload() $\rightarrow$ G_9 representation

Chapter 3

System Analysis and Design

One of the pillars of the *XPRESS* project is the automation present on the bureau level, and it even aims to be a tool for the process planning. As described on section 2.2.2, the configuration experton is the agent responsible for this managing function. The configuration experton is built among four modules: Production Configuration System, Production Simulation System, Production Execution System, and Production Quality System.

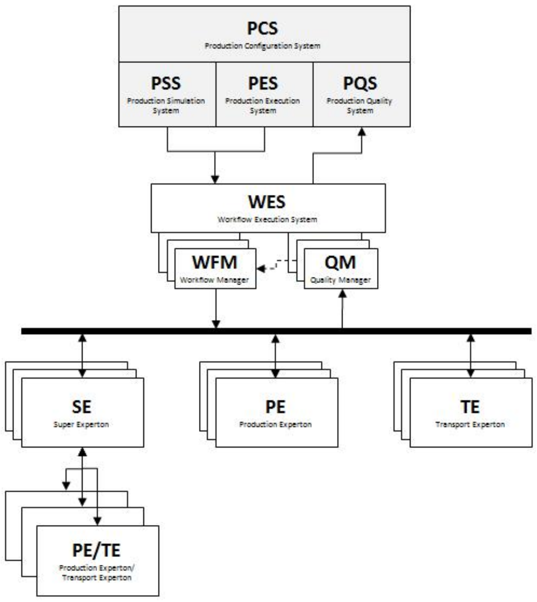

PCS
Production Configuration System
PSS
Production Simulation System
PES
Production Execution System
PQS
Production Quality System
WES
Workflow Execution System
WFM
Workflow Manager
QM
Quality Manager
SE
Super Experton
PE
Production Experton
TE
Transport Experton
PE/TE
Production Experton/ Transport Experton

Figure 3.1: Configuration experton and the interconnections among expertons.

From these modules, the Production Simulation System (PSS) is the responsible for the the creation of new configurations to answer a specific *Job description*. This process is achieved through the aid of simulation tools and by raw data provided by the expertons on the shop floor.

However, it is not possible to simulate every single scenario with the objective of reaching the best configuration.

The work done for this thesis focuses on providing a function library, which will be loaded into the PSS module, with the objective of providing the means to:

- Reduce the complexity of the inicial problem;
- Provide a method for generating solutions;
- Provide a method for comparing solutions.

This library will, from now on, be called ExpertonOptimizationLibrary (EOLibrary).

The tool which will perform the simulations is being developed independently and it is an experton. Therefore it is interfaced using a *TDD* (2.2.1). The simulation boundaries are defined by the engineer.

During the definition of the PSS, the members of the consortium agreed on the structure defined on figure 3.2. This UML model ([17]) defines the interactions between all the modules and it includes the user's inputs.

This sequence is triggered by the arrival of a new Job description from the enterprise's ERP, although some of the functions have to be iniciated by the engineer.

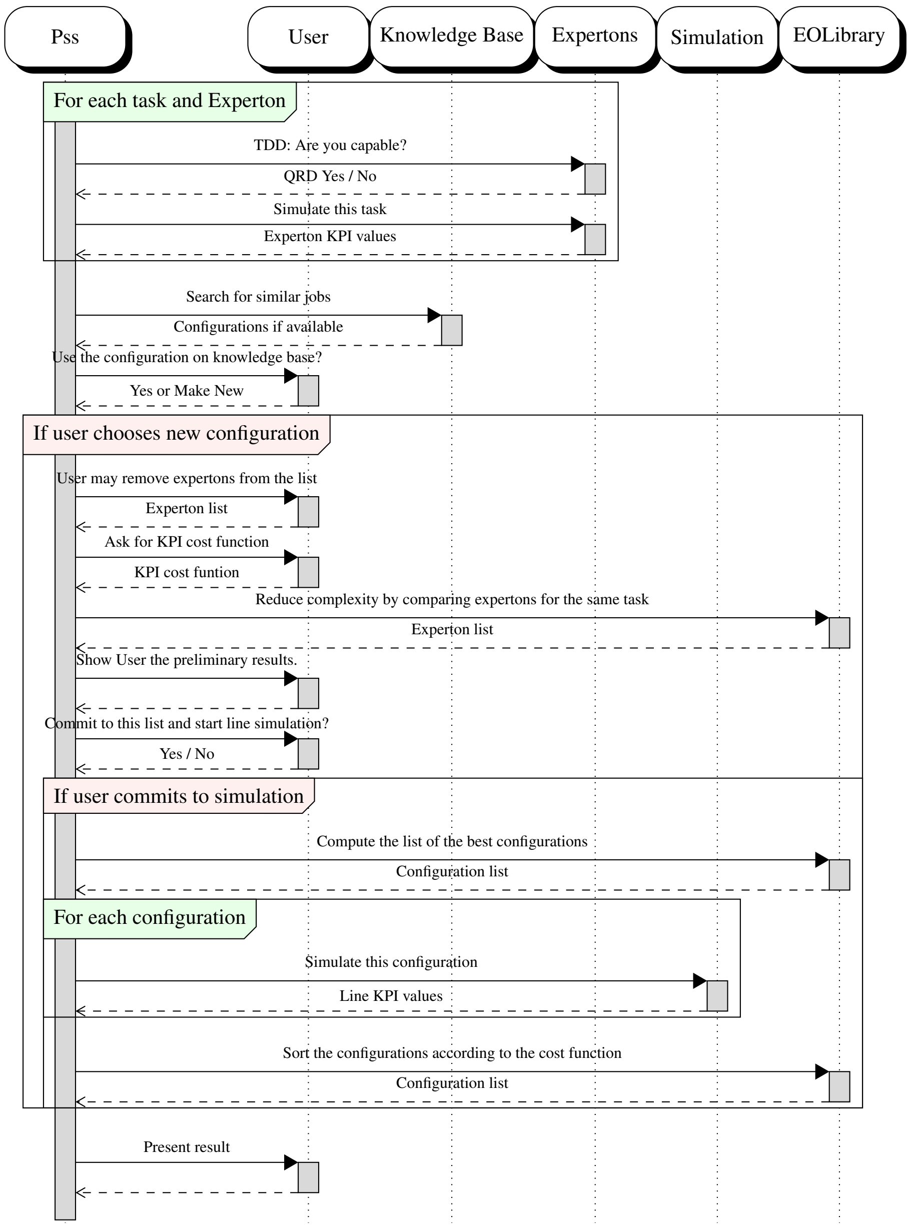

Pss
User
Knowledge Base
Expertons
Simulation
EOLibrary
For each task and Experton
TDD: Are you capable?
QRD Yes / No
Simulate this task
Experton KPI values
Search for similar jobs
Configurations if available
Use the configuration on knowledge base?
Yes or Make New
If user chooses new configuration
User may remove expertons from the list
Experton list
Ask for KPI cost function
KPI cost funtion
Reduce complexity by comparing expertons for the same task
Experton list
Show User the preliminary results.
Commit to this list and start line simulation?
Yes / No
If user commits to simulation
Compute the list of the best configurations
Configuration list
For each configuration
Simulate this configuration
Line KPI values
Sort the configurations according to the cost function
Configuration list
Present result

Figure 3.2: Sequence diagram for PSS.

3.1 The interface structures of the library

The information required and produced by the EOLibrary can be summarized into the following categories.

3.1.1 Job description

The job description is the detailed information that defines the product to be manufactured. It is generated at enterprise's ERP and it is issued to the PCS, arriving ultimately on the PSS. It is from the job description which is extrated the list of tasks to be performed.

3.1.2 Expertons

The PSS is able to retrieve a list of all the expertons available at the shop floor. It is also able to question the experton if it is capable of performing a task defined at the job description and if the answer is positive, it makes an estimation for the individual performance indicators.

3.1.3 The configuration

A configuration is basically an assignment matrix, X, which assigns expertons to tasks.

3.1.4 Key Performance Indicators (KPIs):

The key performance indicators are metrics of performance which chacterize an equipment regarding a given task. These metrics can be:

- Volume related: i.e. throughput (number of units per time unit).
- Quality indicators: yield, rework rate, scrap rate or any metric task/technology dependent.
- Financial indicators: i.e. cost per piece, total cost of ownership.
- Time indicators: i.e cycle time; set up times; productive/downtime (either schedule and unschedule)
- Others.

For a KPI to be useful to the EOLibrary, it must be able to be estimated during experton simulation or during line simulation.

3.1.5 Cost function

To be able to define what is the equipment that fits best, there must be a comparison method. The engineer must provide a cost function. The EOLibrary will, then, apply it whenever a comparison has to be made.

The cost function is a mathematical function of the KPIs that provides a comparable value to each possible solution and, at the end, by choosing the minimum value, it gets the best experton configuration.

The cost function is defined by an array of weights and its application is described on the equation 3.1.

$$CV_{j,k} = \sum_{i} W_i \times KPI_{i,j,k} \tag{3.1}$$

$$CV_{j,k} = \text{Cost value of the experton } j \text{ at the task } k$$

$$W_i = \text{The weight of the KPI } i \text{ on the cost function}$$

$$KPI_{i,j,k} = \text{The value of the KPI } i \text{ takes for the experton } j \text{ at the task } k$$

Finding the best solution is a minimization problem and it is defined by the equation 3.2.

$$min \sum_{j,k} CV_{j,k} \times X_{j,k} \tag{3.2}$$

$$i = \text{KPI index}$$

$$j = \text{Experton index}$$

$$k = \text{Task index}$$

$$X_{j,k} = \text{Element of matrix of assignments. Can be either 0 or 1.}$$

3.2 Scenario constraints

The tools under development are not capable of modelling every factor present on a real situation. Some major factors impose constraints to the scenarios under analysis. The most significant constraints are:

- The layout constraint:

  The layout of the shop floor is considered to be static during all the simulation steps. Therefore, the engineer has the responsability of achieving the best layout configuration beforehand for the process in question. Generally speaking, a layout represents several degrees of freedom and, as a result, the space solution would grow to such a dimention that the simulation tools available would not provide results in an acceptable timeframe. On the other hand, with this approach the layout may restrict the solution on such a way that the optimal point is not even near.

- The Transport Experton constraint: The transport experton manager used to move a certain product part or material from a station to another station is defined by the layout. This means that different transportation scenarios cannot be chosen. On the other hand, the transport experton will affect the line performance and might be a leading factor to the optimal line configuration. Another concern regarding the simulation of the transport expertons is that the simulation does not take into account all the positions that each transport experton may take at the shop floor, but only an average time of availability and an average load of all the transportation system.

These factors come from the fact that the transport expertons may be shared among several lines, so the simulation only takes into account the characteristics of the new line and does not include the other processes that can be run in paralell.

- The simulation constraints: The PSS is able to request simulation results of a production line set up to a modelling tool, through the simulation experton. As the simulation computation can be time consuming, and the space solution has an exponencial growth with the complexity of the production processes and the available list of expertons, there is a need to have a time limit to the whole simulation process, to make the tool suitable for the industry. Assuming the simulation time as the product of the time of a single simulation versus the number of permutations of the available expertons, the simulation time constraint could be used as a threshold for the optimization algorithm which runs before simulation. Or, simply, it could be presented to the engineer before he commits to execute the simulation, leaving him the choice of adding some more constraints to the configuration and start the optimization process all over again.

- Multitasking Experton: This tool will be developed accepting the possibility of an *Experton* being assigned to several tasks in the same process line.

  There might be the case where an experton is fast enough to perform two different tasks. This can be seen by a high standby time on a single task configuration.

  On the other hand, if a scenario with an experton doing several tasks would represent a bottleneck of the line, then, during line simulation, the output of this cell would be poor compared to the other solutions, and its KPIs would be, accordingly, poor.

3.3 Solution space representation

The solution space of configuration definition problem can be represented either by a condensed graph (section 3.3.1) or by the expanded tree notation (section 3.3.2)

3.3.1 Graph of expertons per task

One representation of the initial problem would be as on figure 3.3.

Despite the fact that the graph is not able to represent the restriction to the solutions, it might be a starting point to any algorithm which generates the first list of possible solutions.

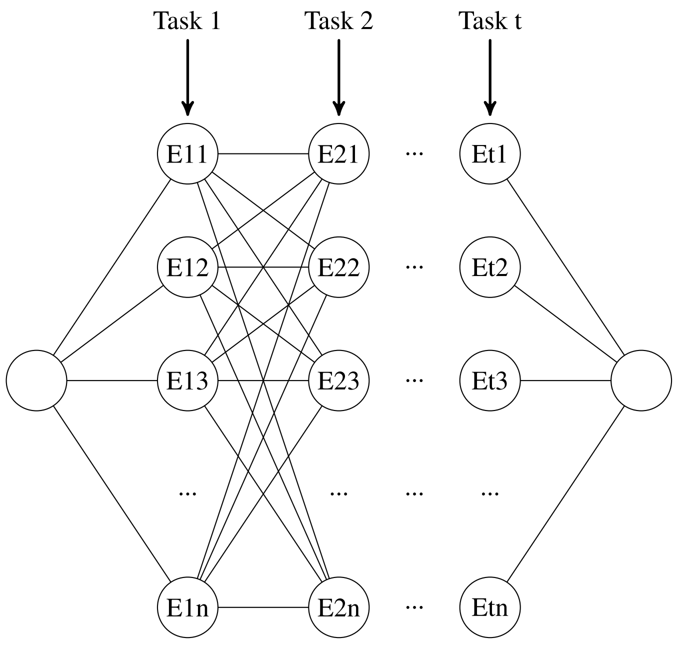


Figure 3.3: Graph for task experton possibilities

3.3.2 Tree view of the solution space

Another type of graphical representation is a tree like graph. When compared to the graph on figure 3.3, it contains more information. Although there is more redundant data and it tends to consume huge memory space when the problem space solution grows.

An example of a tree representation of the solution space is on figure 3.4

Each level of the tree represents a task and, in each level, there is a node for each experton representing each possible path of the following level.

If we represent all the possibilities in a tree based graph:

- The Root is the empty solution.
- Each depth has N_i nodes. Being $N_i = N_{i-1} * E_i$. E_i is the number of Expertons available for the task i.
- The tree has a depth of T, being T the number of tasks to perform.
- A possible configuration is any path connecting the root to a node of the depth T.

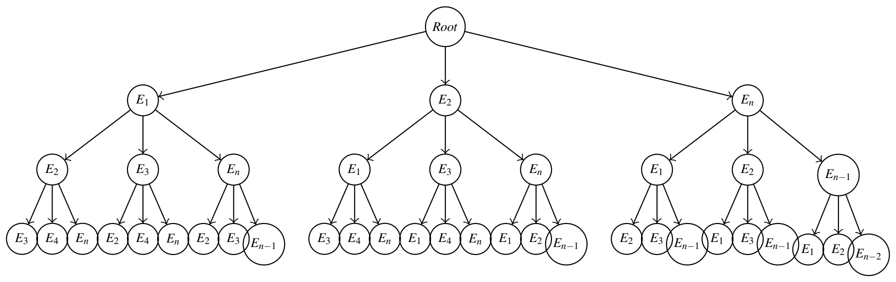

Root
E_1
E_2
E_n
E_2
E_3
E_n
E_1
E_3
E_n
E_1
E_2
E_{n-1}
E_3 E_4 E_n E_2 E_4 E_n E_2 E_3 E_{n-1}
E_3 E_4 E_n E_1 E_4 E_n E_1 E_2 E_{n-1}
E_2 E_3 E_{n-1} E_1 E_3 E_{n-1} E_1 E_2 E_{n-2}

Figure 3.4: Tree view space solution

3.4 EOLibrary class models

The data model which defines the classes implemented by the EOLibrary is described on figure 3.5.

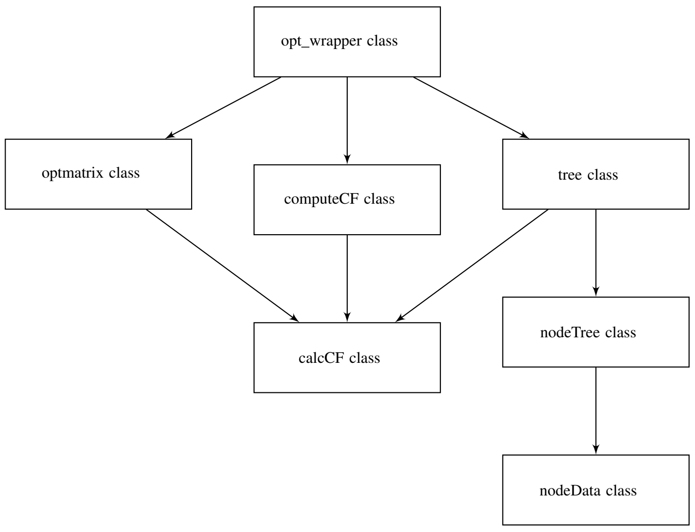


Figure 3.5: Data model of the EOLibrary

3.5 EOLibrary public methods

Before the line simulation of the possible configuration begins, the PSS needs to perform some rough optimisation due to the simulation time restrictions. After the PSS has gathered the list of all the candidate expertons and its estimated KPIs, the space solution is at its peak size. The first measure to reduce the space solution to a more affordable size, has the aim of reducing the number of candidate expertons for each task.

This first optimization will be referred to as ReducedMatrix method. The ReducedMatrix method has the expertons lists and the respective KPIs, and the definition of the cost function as its inputs, and delivers another (the reduced) expertons lists (in the same format as the input).

The second method will analyze, for the first time, the line as a whole. It will not only provide a second optimization but also generate the list of configurations. This method will be referred to as ConfigurationGenerator method. The ConfigurationGenerator method has the expertons lists and the respective KPIs, and the definition of the cost function as its inputs, and delivers the configurations list and the respective cost value.

The last public method makes a comparative analysis of the configurations. It is independent of the ConfigurationGenerator method due to the fact that the line simulation is performed in between. This method will be referred to as ConfigurationRanking method. ConfigurationRanking method has the configurations list and the respective KPIs obtained by the simulator, and the definition of the cost function as its inputs, and delivers the cost value of each configuration.

3.6 Trade-off analysis

On this section, each method will be analyzed regarding its possible implementations.

3.6.1 ReducedMatrix method

At this point, we will look into every task and select the best expertons based on their KPIs (which is the only information available). We can, then, take the following approaches:

- Calculate the cost function for each experton, and then select the best group.
- Or start by applying a cross-reference measure (described at 2.3.3), and then apply it to the cost function.

The latest option has the ability to make a comparison, which is independent of the weights of the cost function defined by the engineer, and, in this case, striving for a pareto optimal solution.

The cross-reference measure cannot be used alone, for any experton with the best KPI individually would have the same effective efficience (equal to one).

The impact of this algorithm is highly dependent on the initial and final number of candidates. Therefore, the probability of eliminating a good solution overall, by removing a bad candidade at a given task, increases as the inicial number of candidates decreases. To counterweight this fact, the number of expertons removed from the candidate list will be logaritmically dependent on the inicial size.

The equation will then be described on the implementation chapter (chapter 4).

3.6.2 ConfigurationGenerator method

At first sight there might be the temptation of applying the assignment algorithm and retrieve the best solution. However, the KPIs that are available are not obtained using real data but using algorithms which produce rough estimations. Moreover, at this point, the line simulations have not been performed yet, so data, such as, the impact of the production of one experton into the production of the following tasks is still missing. To conclude, the configuration generation algorithm needs to provide the most broadened list as possible, but, as it is impossible to provide every single configuration, elimination should provide the lowest risk possible.

If we want to use the assignment algorithm to generate the best n solutions, the algorithm has to run n times and it has to have a method to increase the weight of an already found solution, finding each time a new one.

Another alternative, is to create an algorithm based on the tree represented on figure 3.4. However, traditional algorithm to manipulate trees go through the nodes in a sequencial way, and, consequently, they are time and memory consuming. They can be one of the following type:

- *Breadth-First (BF).* The BF goes through all the nodes of a level before it enters the next one.
- *Depth-First (DF).* The DF goes through all the child nodes of a node X before it moves to the next one of same level of X.

Both these alternatives can cover the whole tree, and the one which performs best only depends on how the data is stored onto the memory and on how fast it is to access the following element. These are brute force algorithms, in the sense that they need to test every configuration before they can assess its ranking, and there is no risk of eliminating a good solution.

In this case, we need a more *intelligent* tree progression mechanism. Some candidates are the following:

- *Best First*, which explores a tree by expanding the most promising node chosen according to a specified rule. The node is characterized not only by its weight, but also includes the weight of its parent node. This can be summarized as the weight of the path until the node.
- *A Star (A*)* as a special case of a Best first. It takes into account the weight of the path until the node and an estimation of the weight of the path ahead. In fact, if the estimation until the goal is zero for every node, this algorithm will behave as the Dijkstra's algorithm.
- *Hill climbing*, which is an algorithm that compares a node with its neighbours to get the steepest ascent. The drawback of this algorithm is that it only finds the local maximum, leading to possible incorrect assumptions. In our case, we cannot guarantee that the graph converges to only one maximum.
- *Quick sort* based selection algorithm (kbest algorithm).

When using the Best First algorithm to generate the k best solutions, it tends to behave as a Breadth First at the levels near to the root of the tree, and its full potencial only apears on the rest of the tree. This can easily be seen because, as it gets deeper into the tree, the path weight becomes so heavy, that the next best node is the one at the highest level, even if the node itself is much worse than its neighbours.

It is in this characteristic that the A Star algorithm can be beneficial. As A Star includes an estimation of the weight of path, when compararing nodes of different levels, their value has a more similar meaning, and they are then more comparable.

In this particular case, an adaptation of the A Star algorithm was used. The adaptation comes on two facts. The first one regards the weight estimation: instead of having an estimation function for the path, we used the weight of the child node with the minimum weight. This information is easily available prior to the start of the generation of the tree, as it is obtained by calculating the cost function of each experton in each task, and then by selecting the minimum and, finally, by adding their values. It is also possible to define, how many *future* levels (tasks) should be taken into account for a corresponding weight.

Additionally, some kind of divide and conquer approach can also be used. This is possible if there are tasks that are completely indepentent from each other. However, this is difficult to put in practice because there is no way of identifying the criteria for a safe division. Therefore, such algorithm is most likely unfeasable.

3.6.3 ConfigurationRanking method

After simulating all the possible scenarios, there is only the need to calculate the cost function for each configuration with the KPIs returned by the line simulation, which should be more reliable than individual experton simulation, and then rank the results onto a list. The application of the cost function solves the multicriteria decision problem, obtained by the fact that there is a list of configurations, each of them characterized by a different set of performance values.

Chapter 4

Development of the library

This chapter describes the work related with the development of the library source code. The programming language used was the C# [18] and the Microsoft Visual Studio 2008 as develop environment tool. The chapter focuses on the implementation decisions and on the complexity analysis.

The EOLibrary interface is composed by the three methods described on section 3.5. The respective syntax is the following:

- The ReducedMatrix method is called through the function:
  static int[ ][ ] **first_opt** (int[ ][ ] matrix, float[ ][ ][ ] KPIs, int[ ] costfunction)

  ← ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to tasks. The second dimension is indexed to expertons. The value is the experton id.

  ← ***KPIs*** is a 3 dimensional array argument. The first dimension is indexed to tasks. The second dimension is indexed to expertons. The third dimension is indexed to KPIs. The value is the KPI measure.

  ← ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

  Returns: A 2 dimensional array. Its stucture is the same as matrix.

- The ConfigurationGenerator method is called through the function:
  static void **second_opt** (int[ ][ ] matrix, float[ ][ ][ ] KPIs, int[ ] costfunction, out int[ ][ ] confs, out float[ ] funcaovalores)

  ← ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to tasks. The second dimension is indexed to expertons. The value is the experton id

  ← ***KPIs*** is a 3 dimensional array argument. The first dimension is indexed to tasks. The second dimension is indexed to expertons. The third dimension is indexed to KPIs. The value is the KPI measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

$\rightarrow$ ***confs*** is a 2 dimensional array. The first dimension is indexed to configurations. The second dimension is indexed to tasks. The value is the experton id.

$\rightarrow$ ***funcaovalores*** is an array. The array is indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].

- The ConfigurationRating method is called through the function:
  static float[ ] **third_opt** (float[ ][ ] confs, int[ ] costfunction)

$\leftarrow$ ***confs*** is a 2 dimensional array argument. The first dimension is indexed to configurations. The second dimension is indexed to KPIs. The value is the KPI measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

Returns: A array indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].

4.1 Methods implementation

4.1.1 ReducedMatrix method

The ReducedMatrix method receives, as input, the matrix which associates experton candidates to a task, the matrix with the KPIs associated to each element of the matrix of expertons and the weight of each KPI according to the user's preference.

It starts by calculating the cross-reference efficiency of each experton compared with the expertons which share the same task.

The value of this efficiency, E_{kk}, is not a good metric to compare the expertons by itself. However, it is a good factor to add to the analysis. Therefore, when comparing several expertons to perform a certain task, the cost value, obtained by applying the cost function to its KPIs, is multiplied by a factor dependent on the efficiency value.

Due to the fact that the cross-reference is based on a maximization problem and the choosing expertons, a minimization problem, the values of the KPI matrix have to be manipulated. This manipulation consists, basically, on replacing the element of the matrix by the result of substracting of the value of the element from the value of the worst (the highest value) experton for that particular KPI

On the other hand, the efficiency value E_{kk} cannot be used directly due to the same reason. Therefore, the factor $\frac{1}{E_{kk}}$ is used instead.

The final aspect regarding this algorithm is the function which determines the number of expertons to return to the PSS.

$$Number\ of\ expertons = (integer)1 + 5.3 * \log 10(Inicial\ number\ of\ expertons) \tag{4.1}$$

This equation, 4.1, was obtained considering the following factors:

- There should be no optimization at this point, if a task has three candidates or less.
- The reduction of expertons should be gradual for tasks with a number of candidates between four and ten.
- And finally, there should be a severe reduction for a number of candidates higher than ten.

Table 4.1 shows some examples of values for the final number of expertons.

Inicial No	Final No
1	1
2	2
3	3
4	4
5	4
10	6
20	7
30	8
40	9
50	10
100	11
200	13

Table 4.1: Table with final number of expertons function

4.1.2 Cross-reference efficiency algorithm

The implementation of the cross-reference efficiency measure, as described at the section 2.3.3, was achieved through the usage of a public library called GNU Linear Programming Kit (GLPK) [19].

This library provides functions for solving linear programming problems based on the simplex algorithm. The information that is needed for the function is :

- The set of variables and their characteristics:
  - Name;
  - Type of boundaries (free, variable with lower bound, variable with upper bound, double-bounded or fixed variable) , low and high values;
  - Coeficient of its weight on the objective function;
  - Kind of variable (continuous variable, integer variable or binary variable);
- The set of conditions and their characteristics:
  - Name;

    - Type of boundaries (free, variable with lower bound, variable with upper bound, double-bounded or fixed variable) , low and high values;

- The coeficients matrix, which associates each variable to each condition.

The algorithm, which implements the cross-reference efficiency measure, creates:

- The number of expertons times the number of KPIs variables. The weight of each variable for the objective function depends on the experton. If it is the experton on analysis, it takes the value of its KPIs, otherwise, it is zero.

- The square number of expertons conditions.

- The coeficient matrix.

Then, the glpk library functions are called and the objective value is retrieved, which, in this case, represents the cross-reference efficiency of an experton within a certain task.

4.1.3 ConfigurationGenerator method

The ConfigurationGenerator method is implemented having a tree structure as its core. The class which defines this stucture includes both the means to store its data nodes, and the generation algorithm.

4.1.3.1 The Tree class

The tree class is a data structure where data can be stored on nodes which additionally have relations with other nodes in a tree like graph.

The tree class also implements all the methods for the manipulation of this data structure.

Each node is a nodeTree element.

The data model of the nodeTree is described on figure 4.1

All nodeTrees, except the root node of the tree, have a valid nodeData (described on figure 4.2), which is the combination of an experton and a task.

And all the nodeTrees of a certain depth belong to the same task.

The connection with the nodes is performed via two double linked lists. One horizontally, aggregating nodes which share the same parent node, and one vertically, aggregating the parent and its first child.

4.1.3.2 Tree generation Algorithm

The algorithm which creates de tree basically performs five operations:

1. It calculates the weight of each node to the solution.

2. It calculates the estimated weight for the A Star algorithm for each task.

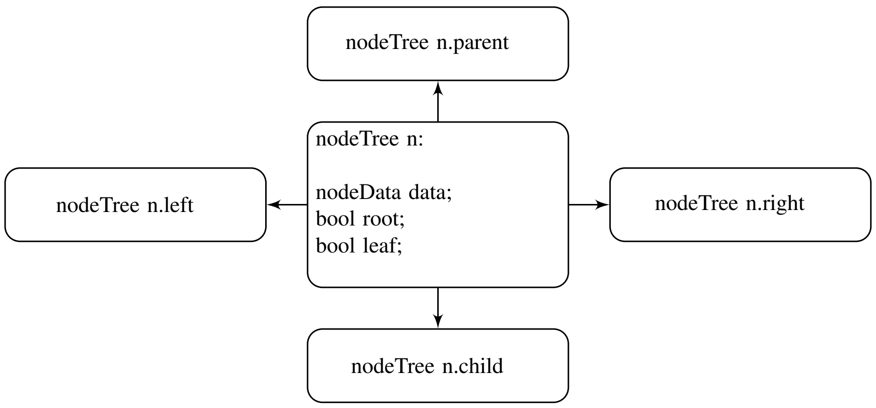


Figure 4.1: nodeTree data model

3. It adds the dummy root node to the tree and adds all the children from the first task into the node priority queue.

4. It fetches the first item of the node priority queue and inserts it to the tree.

5. It adds all it child nodes to the node priority queue.

6. It goes back to point number 4 until there are no more elements at the priority queue, or the condition of the number of solutions equals the number of leaf nodes.

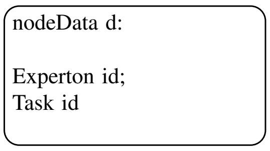
7. It removes all unnecessary nodes from the tree.

The node priority queue is a data structure which stores all the possible next nodes to be inserted on the tree. A node is inserted onto this structure according to a weight. Is is implemented using an Insert Sorted kind of algorithm, as it is less time consuming when compared to other sort or search methods.

As in each cycle one node is removed from the queue, and possible several nodes are added, this queue has an undercontrolled expantion until only leaf nodes are present or the algorithm termination criteria has been reached. This may represent an unmanageable situation, when the insertion operation is no longer a constant time operation - due the the overhead of manipulating huge memory structures, leading to long processing times.

To prevent such a scenario, the queue has a maximum size. When this size is reached, a portion of the queue is removed.

The size of the queue should be as big as possible until it becomes unsustainable. On the other hand, although the queue is reduced by eliminating its worst elements, it is not risk free. For example removing elements in early phases might eliminate better solutions than the obtained at the end of the process. So the portion which is eliminated should be as small as possible. However, the execution time should decrease as the portion of the elimination increases, as it will decrease the number of eliminations, and consequently, the time spent on the release of the memory space from the queue.


nodeData d:

Experton id;
Task id

Figure 4.2: nodeData data model

The weight of a node, depends not only on its own KPIs but also on the aggregate weight of the path from the root until that node. One implication to this weight computation is that all KPIs are treated as additives. There is no way to distinguish an additive from a product based KPI.

The algorithm has an additional parameter (Depth parameter) which allows us to include an estimation of a weight of the future child nodes. The aim of adding future weight to a node is to make the algorithm explore the most promissing branch of the tree first. Forthermore, in the leaf nodes, the weight is the same as using no prediction method, which allows us, in the end, a true comparison between solutions.

The depth parameter defines how many consecutive levels have to be estimated. The estimation itself is calculated using the minimum KPIs of the next task. This ensures that no node is penalised when compared to other on the same task. But, at the same time, it helps to decrease the priority of a bad node which belongs to a earlier task.

Pratical numbers for all parameters are presented further onto the chapter.

Another note regarding this algorithm is that, if:

- The number of solutions requested is higher than the number of combinatorial possibilities;
- There is no elimination on the priority queue;

Then the results produced are equivalent to any kind of brute force algorithm, as it has eventually run into all the nodes. It is also independent of the depth parameter, because it does not have influence over the leaf nodes, which represent ultimately the weight of the whole path.

4.2 Algorithm complexity Analysis

On this section, the behaviour of the algorithms and its data structures will be analysed regarding their estimated impact over the memory usage and the run time.

4.2.1 ReducedMatrix method

The algorithm is processed once for each experton at each task. Therefore, the run time is mostly dependent on the duration of the simplex algorithm, employed to retrieve the cross-reference efficiency, times the total number of candidates for all tasks, which is given by equation 4.2.

$$\text{Total number of candidates} = \sum_{i=1}^{\text{Number of tasks}} \text{Number of candidates at task } i \tag{4.2}$$

This algorithm, then, has a linear time complexity, $O(N)$. The memory complexity is linear as well, due to the fact that the working array, which stores the expertons, has a size linearly dependent on the total number of candidates.

4.2.2 ConfigurationGenerator method

To analyze the complexity of the ReducedMatrix method, the major components of the algorithm have to be accessed first.

The first major component is the priority queue. The operations over a queue provide constant time ($O(1)$) [20]. However each time an element is inserted into the queue, the position is calculated using a binary search algorithm that executes in $O(log(N))$. This the the minimal penalty for implementing the priority. Other solutions would require $O(1)$ for the insertion time, but the consequently sorting function would run at least on $O(N)$ time.

As the number of insertions into the priority queue is potentially higher than the number of remotions, it would be logical to implement the insertion method with run time $O(1)$ and give the burden of execution to the removal method. This , however, is not beneficial, as the best searching algorithm for an unsorted array takes $O(N)$ time and the insertion/removal ratio is not high enough to compensate.

The second component for the execution time analysis is the tree. The insertion time of a new node into the tree is $O(1)$ based and there is no other time consuming operation during the creation of the tree.

The sum of all nodes of the tree with all nodes of the queue is less or equal to the maximum number of nodes of a problem. This is, in fact, a concern: due to the exponential characteristic of the number of nodes for a combinatorial problem, there is the risk of reaching the limit of memory space provided by the operating system.

4.2.3 ConfigurationRanking method

The analysis of the ConfigurationRanking method can be directly inferred from the analysis of the ReducedMatrix method. In fact, the comparison of line configurations performed on this method is analogue to the comparison of expertons on the ReducedMatrix.

4.3 Distribution and usage of the EOLibrary

The library was developed on the C# language, as previously stated, and compiled into a Dynamic-link library (DLL). The inclusion into other projects is done through the option "Add reference" of the development tool. This way, a project under development is not only capable of calling the public methods of the library but the development tool can also use the original source code for syntax checking purposes and debug functionality. Finally, the GLPK library provides a C based DLL. Because of this, it must also be included the DLL which converts the structure entities between the two languages. This final set o DLL's must be available at the same folder as the executable file of the application.

Chapter 5

Functional Testing and Characterization

One of the main concerns when implementing optimization algorithms is that some of the best solutions may get scraped and, consequently, a worse solution is kept instead.

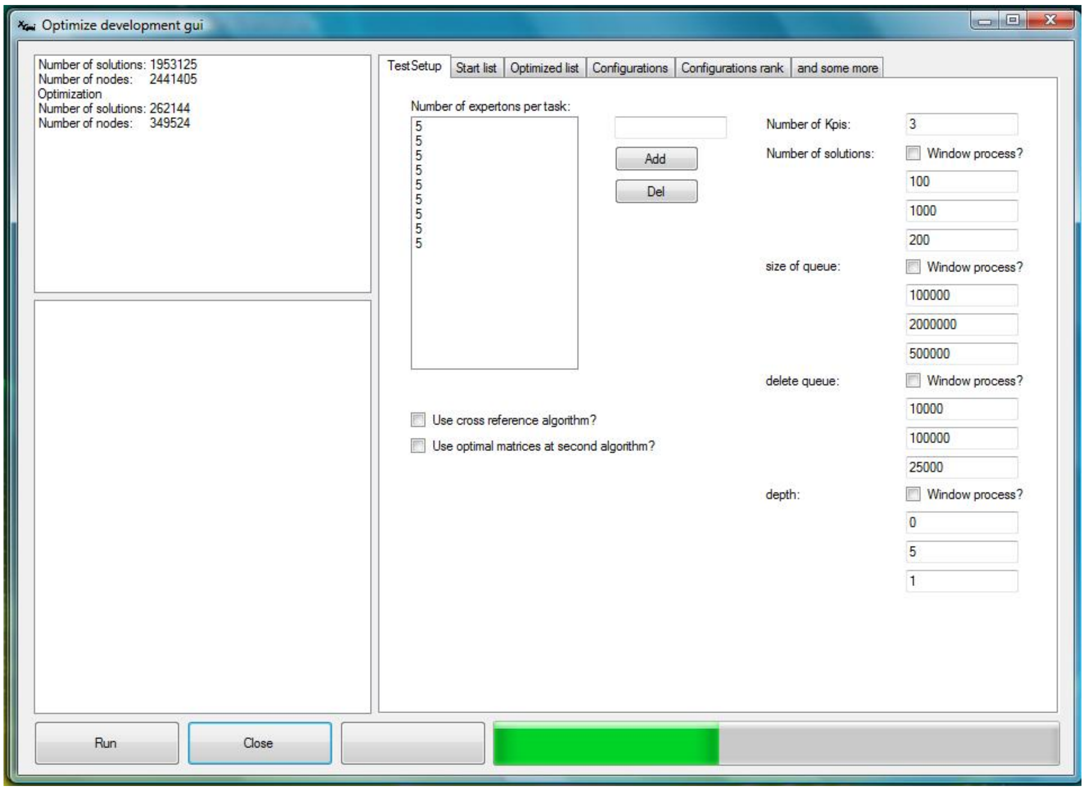
To measure the probability of not eliminating of a good solution, named yield afterwards, a battery of tests was performed. Those tests were meant to simulate a broad list of scenarios with random KPIs.

A graphical user interface (figure 5.1), independent of the *XPRESS* project, was developed having in mind the following characteristics:

1. To be able to perform the first testing of the library interface.
2. To be able to measure the performance of the algorithms.
3. To be able to compare solutions against a brute force method.

The first characteristic has a debug purpose, while the combination of the last two characteristics is the key for the analysis provided by this report.

The tool is able to generate scenarios of any size, and will populate the KPIs with random numbers. At this point, the existance of real data is not relevant to the behaviour of the algorithms.



```
Number of solutions: 1953125
Number of nodes:    2441405
Optimization
Number of solutions: 262144
Number of nodes:    349524
```

Number of expertons per task:
5
5
5
5
5
5
5
5
5

Number of Kpis: 3

Number of solutions: ☐ Window process?
100
1000
200

size of queue: ☐ Window process?
100000
2000000
500000

delete queue: ☐ Window process?
10000
100000
25000

depth: ☐ Window process?
0
5
1

☐ Use cross reference algorithm?

☐ Use optimal matrices at second algorithm?

Figure 5.1: Screenshot of the development interface.

5.1 Machine characterization

Measuring the time which an algorithm takes to run depends, highly, on the system characteristics in which it runs.

All measurements were using a machine with the following characteristics:

- Processor: Intel Core2 Quad Q9300 @ 2.5GHz
- Memory: 4GB
- Operating System: Windows Vista Business N

As the algorithm is completely defined inside the same thread and the fact that this machine has a multicore processor, makes the time measurements less dependent on other processes that may be running in parallel.

5.2 ReducedMatrix method

Using this tool, two scenarios were set up, both aimed to test the first optimization. The interface with the library was also temporarily changed to allow different approaches to be tested.

The first scenario was a Job description with only one Task and twenty Experton candidates to the task. This scenario had the intention to prove that the cross-efficiency metric was correctly implemented and, in this case, it would provide the same results as simple comparison of the KPI value.

The input data is represented on table 5.1 and the respective results, on table 5.2. As expected, the results show that the cross-reference is able to order a list of expertons correctly.

Experton Id	KPI value
1	54
2	30
3	32
4	54
5	39
6	45
7	66
8	38
9	60
10	32
11	38
12	41
13	62
14	67
15	57
16	54
17	52
18	53
19	59
20	47

Table 5.1: KPIs generated for the first scenario

Experton id	2	10	3	11	8	5	12
KPI value	30	32	32	38	38	39	41
E_{kk}	1	0.917	0.917	0.667	0.667	0.625	0.542

Table 5.2: Results of the first scenario

The second scenario had the aim of checking the impact of the cross-reference measure on the resultant experton list. This scenario was composed by one task, twenty experton candidates and three KPIs to characterize the experton.

With the data generated, and presented on table 5.3, the results, presented on table 5.4, show that cross-reference efficiency measure improves the ranking of expertons which are the best at, at least, one KPI. One example of this is the Experton with id 7.

The time consumed by this algorithm is neglegible.

Experton Id	KPI 1	KPI 2	KPI 3	E_{kk}
1	58	54	50	0.441
2	35	49	51	1.000
3	53	62	64	0.192
4	46	36	64	0.890
5	60	59	60	0.128
6	36	59	68	0.867
7	45	60	31	1.000
8	40	36	35	1.000
9	61	35	33	1.000
10	32	61	62	1.000
11	56	60	48	0.485
12	65	65	45	0.576
13	51	49	59	0.463
14	64	42	45	0.672
15	65	55	31	1.000
16	47	31	37	1.000
17	66	47	62	0.484
18	55	44	56	0.581
19	55	47	48	0.541
20	45	44	43	0.724

Table 5.3: KPIs generated for the second scenario and the resultant efficiency measure

Experton id	8	16	9	20	2	4	17
Cost value	111	115	129	132	135	146	155

Experton id	8	16	9	2	7	4	10
$\frac{\text{Cost value}}{E_{kk}}$	111.000	115.000	129.000	135.000	136.000	146.099	155.000

Table 5.4: Results of the second scenario, without and with cross-efficiency impact

5.3 ConfigurationGenerator method

The second optimization algorithm has several parameters which may influence the solution and the performace characteristics.

In order to be able to identify the impact of each parameter, specific scenarios were defined at the graphical interface.

The parameters available, that control the behaviour of the A star algorithm, are: the number of solutions to generate; the depth at which the A star algorithm should look into the tree for the estimation weight; the maximum size of the priority queue; and the size of the portion of the priority queue that is deleted, when it reaches its maximum size.

For a better comparison, independently of the number of configurations generated, only the best 100 configurations were then checked their position on the ranking of a brute force algorithm. If the configuration is also found at the best 100 configurations of the brute force algorithm, then it is considered to be a positive match. The number of positive matchs will give the probability, given a set of conditions, for a configuration to belong to, in fact, the best "100" configurations overall. This probability will be refered to as the yield of the algorithm.

The brute force algorithm is achieved using the A star algorithm, but with the condition that: the number of solutions to generate is higher than the number of combinatorial possibilities; and the size of the priority queue is higher than the maximum number of nodes the graph.

The first scenario aimed to define the impact on the number of configurations which should be generated and the depth at which the A star algorithm should look into the tree.

In order to do this, a scenario, with only 8 tasks and with 5 expertons per task, was set up. This scenario has $5^8 = 390625$ combinatorial configurations. The tree representing this screnario has 488281 nodes. Therefore, if the size of the priority queue is at least 488281, then, for sure, there will be no deletion at the queue.

Table 5.5 shows that the yield grows as the number of solutions generated increases. This was the expected behaviour. A first conclusion is to request the algorithm at least ten times the desired number of configurations for a good operating point.

It should also be mentioned that, the brute force algoritm took 30.4 minutes to run. The time spent by all the parameterizations is on table 5.6. A huge time saving can already be observed. This scenario was in fact very small to make true measurements of the time spent. However, the correlation between the time and yield starts to be noticed.

A second scenario was used, this time to evalute the impact of the deletion of the poor end of the priority queue whenever it gets full.

This scenario is composed by a Job description of ten tasks, and each task has 5 experton candidates. This scenario has $5^{10} = 9765625$ combinatorial configurations. The tree representing this screnario has 12207030 nodes.

For this scenario, the brute force algorithm was relaxed on the number of solutions it should generate. The fact of the brute force algorithm is also implemented by the A star algorithm should be taken into account, otherwise this simplification could not be made. Nevertheless, the number

Yield (%)	Number of solutions generated							
depth	100	200	300	400	500	700	900	1000
0	44	72	92	98	100	98	98	98
1	32	49	65	73	80	89	100	100
2	32	49	65	73	80	89	99	99
3	32	49	65	73	80	89	99	99
4	32	49	65	73	80	89	100	100
5	32	49	65	73	80	89	100	100
6	32	49	65	73	80	89	100	100
7	32	49	65	73	80	89	100	100
8	32	49	65	73	80	89	100	100

Table 5.5: Variation of the conditions of the first scenario.

of solutions requested for the brute force condition was 100 000. This is one hundred times what was requested for all the other conditions.

Despite the fact that of only one percent of the solutions were generated, the program needed 75 minutes to complete the procedure.

Moreover, the results from the previous scenario support this approach. Therefore, the risk of data being deceiving is low.

From tables 5.7 to 5.9 cannot be concluded any trend, specially, regarding time.

At this point, we can assume that for problems with the size of the previous scenarios, the following conditions, will also lead to good results (high yield):

- The number of solutions gathered equals 1 000, then select the 100 best.
- The size of the priority queue equals 2 000 000 without penalizing the execution time.
- The deletion size equals 50 000.
- And the depth parameter equals 0.

Now, to evalute the *shape* of the problem, four scenarios were setup. These scenarios only differ in the order in which the tasks are placed along the Job description. Moreover, each task has a different number of expertons, represented on table 5.10.

Although the number of combinatorial possibilities is the same for all the scenarios, equal to 725760, the number of nodes to represent the whole tree differs quite considerably. As can be seen on table 5.11, the third scenario (the one with the highest number of nodes) has more 54% of nodes compared to the forth scenario (the one with the lowest numer of nodes).

The results, available on table 5.12 show that a best first algorithm provides more reliable results when at the extremities of the tree have the least amount of candidates. An interesting point can be seen on the chart 5.2. On the fifth scenario, the number of nodes at each level of the tree (n) follows the geometric series ar^n with $a = 0.2254$ and $r = e^{1.9433}$.

So far, all the comparisons between scenarios had the same number of tasks. For this reason, another test case was provided to compare the impact of number of tasks on the results. Two new

Time (seconds)	Number of solutions generated							
depth	100	200	300	400	500	700	900	1000
0	01.0	01.0	01.0	01.0	02.0	02.0	02.0	02.0
1	01.0	01.0	01.0	01.0	01.0	02.0	02.0	02.0
2	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
3	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
4	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
5	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
6	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
7	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
8	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0

Table 5.6: Time consumption for each optimization call of the first scenario (seconds)

scenarions were set up, both with the same amount of combinatorial possibilities. The seventh scenario has 12 tasks composed by 3 experton candidates each (a total number of nodes of 797161), while the eighth scenario has 6 tasks composed by 9 experton candidates each (a total number of nodes of 53144).

Based on the results on table 5.13, it can be concluded that the best first algorithm can obtain faster results for problems with less tasks. However, the reliability of the results is penalized in this case.

The fact that the run time of the brute force algorithm took quite similar time and the fact that there is a difference of 25% on the number of nodes can only be explained by differences in the access time to the memory. In fact, the thinner the tree, the higher the probability of nodes at the same level having similar weights. Then, the serialization of the nodes into the priority queue will be more straightforward. Another remark is the fact that on the seventh scenario, for each node that is removed from the priority queue, only three new nodes are inserted. This shows that the priority queue will be, at all times, with a less quantity values, turning the insertion operation faster.

To conclude the results of the tested scenarios, on table 5.14 is summarized the influence of the variation on the algorithm parameters on its performance measures.

Finally, on table 5.15 is summarized the influence of the problem definition into de performance measures of the algorithm.

Yield (%)	Size of the deletion		
Queue size	10000	35000	60000
1000000	100	100	100
1500000	100	100	100
2000000	100	100	100

Table 5.7: Results from the second scenario - variation of the size of the queue and the size of the portion to apply the deletion. The depth parameter took the value zero.

Yield (%)	Size of the deletion		
Queue size	10000	35000	60000
1000000	95	95	95
1500000	95	95	95
2000000	95	95	95

Table 5.8: Results from the second scenario - variation of the size of the queue and the size of the portion tho apply the deletion. The depth parameter took the values from one to three.

Time (seconds)	Size of the deletion		
Queue size	10000	35000	60000
1000000	198.3	194.0	209.7
1500000	204.7	167.7	181.3
2000000	189.3	170.0	164.7

Table 5.9: Time consumption for each optimization call of the second scenario (seconds)

	Scenario			
Task No	3	4	5	6
1	10	2	2	10
2	9	3	4	8
3	8	4	7	6
4	7	6	9	2
5	6	7	10	3
6	4	8	8	4
7	3	9	6	7
8	2	10	3	9

Table 5.10: Distribution of candidates per scenario (from 3rd to 6th)

	Scenario			
	3	4	5	6
No Nodes	1245701	807585	1013611	822331

Table 5.11: Number of nodes per scenario (from 3rd to 6th)

	Scenario			
	3	4	5	6
Execution time of the brute force model (hh:mm:ss.sss)	01:53:25.880	01:30:35.712	01:29:17.173	01:25:31.364
Execution time (hh:mm:ss.sss)	00:00:37.314	00:00:00.983	00:00:11.840	00:00:02.137
Yield at 100 solution(%)	100	98	99	100
Yield at 1000 solution(%)	57.0	39.5	84.4	27.3

Table 5.12: Summary of the results per scenario (from 3rd to 6th)

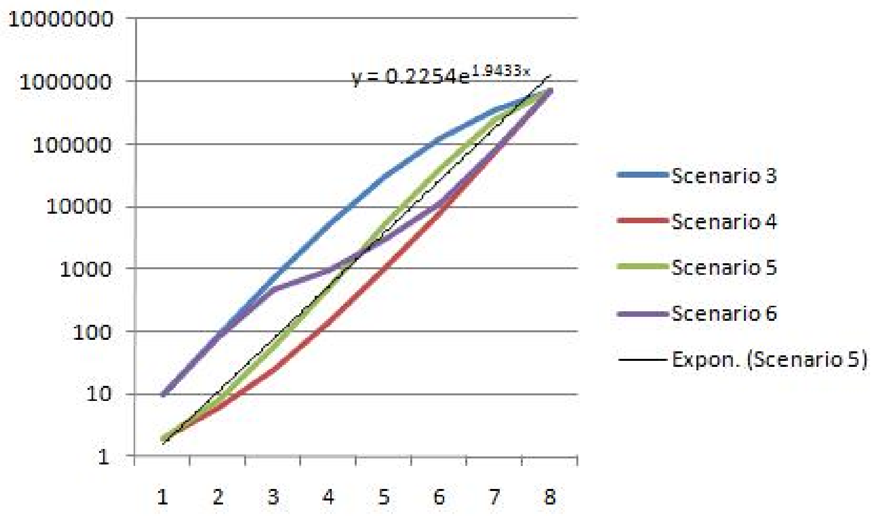


Figure 5.2: Chart with the number of nodes for each scenario (from 3rd to 6th). Y axis with logaritmical scale.

	Scenario	
	7	8
Execution time of the brute force model (hh:mm:ss.sss)	00:49:35.474	00:50:25.775
Execution time (hh:mm:ss.sss)	00:00:10.810	00:00:01.279
Yield at 100 solution(%)	99	100
Yield at 1000 solution(%)	88.2	63.0

Table 5.13: Summary of the results per scenario (7th and 8th)

Parameter	Variation	Influence on Yield	Influence on Run Time
Number of solutions	$\nearrow$	$\nearrow$	$\nearrow$
Depth	$\nearrow$	$\searrow$	$\searrow$
Queue size	$\nearrow$	$\rightarrow$	$\searrow$
Deletion size	$\nearrow$	$\rightarrow$	$\searrow$

Table 5.14: Influence on Yield and Run Time due to variation on the parameters

Condition	Number of nodes	Influence on Yield	Influence on Run Time
More candidates at the first tasks	$\uparrow$	$\nearrow$	$\uparrow$
More candidates at the last tasks	$\downarrow$	$\searrow$	$\downarrow$
More candidates at center tasks	$\nearrow$	$\uparrow$	$\rightarrow$
More candidates at extreme tasks	$\searrow$	$\downarrow$	$\searrow$

Table 5.15: Influence of the problem definition into the Yield and Run Time

5.4 ConfigurationRanking method

The ranking of the configuration, based on the KPIs provided by the simulation, has the same algoritmical characteristics of the first optimization. Therefore, no scenario was tested as its conclusions would be redundant.

Chapter 6

Integration and Functional Sample

This chapter describes the integration of the developed library into the software provided by another member of the consortium.

6.1 Integration

The optimization functions were delivered packed into a library (.dll) to be used by the PSS software.

The PSS user interface can be seen on the screenshots of figures 6.1 to 6.6

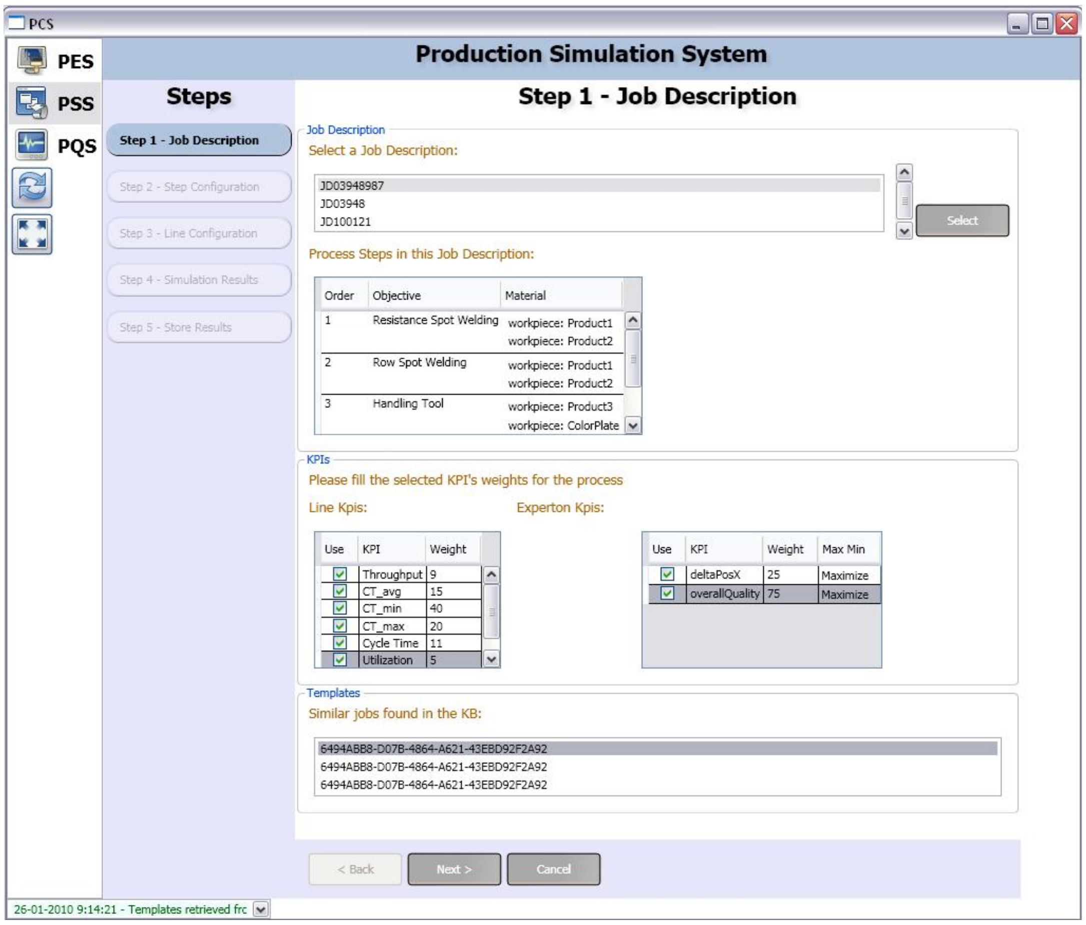
The several steps of the graphical user interface correlate with the several points of the sequence diagram described on figure 3.2. However, it masks some functional steps which is run on background processes.

When an engineer (PSS user) loads the PSS software, he is presented with the list of *Job description* available at the database (which were created by the ERP software). When the user selects the job description, he can decide to use a configuration which already exists on the knowledge base, or fill in the KPI cost functions weights for the simulation phases. This information is visible on figure 6.1.

On step two, figure 6.2, the user can restrict the list of expertons by selection. As the interface with the expertons (including the simulation experton), even for simulation purposes, is through the usage of *TDDs*, the PSS will generate a template *TDD* with the minimal required information. The user can, at this point, edit the *TDD* to include additional parameters. This task is beneficial to the simulation, as the prediction of the KPIs becomes more precise. Nevertheless, there might be many cases, on which is not practical to customize all the TDDs.

On step three, figure 6.3, the list of configurations sent to the simulation experton is presented. Once more, the user can select or remove individual elements of the list, and, he can also change the simulation parameters by editing the simulation *TDD*. At the same time, if it is of the user's interest, he can review, in detail, any configuration (as on figure 6.4).

On step four, figure 6.5, the raw results from the simulation are presented.



Production Simulation System

Step 1 - Job Description

Job Description

Select a Job Description:

JD03948987
JD03948
JD100121

Process Steps in this Job Description:

Order	Objective	Material
1	Resistance Spot Welding	workpiece: Product1 workpiece: Product2
2	Row Spot Welding	workpiece: Product1 workpiece: Product2
3	Handling Tool	workpiece: Product3 workpiece: ColorPlate

KPIs

Please fill the selected KPI's weights for the process

Line Kpis:

Use	KPI	Weight
✓	Throughput	9
✓	CT_avg	15
✓	CT_min	40
✓	CT_max	20
✓	Cycle Time	11
✓	Utilization	5

Experton Kpis:

Use	KPI	Weight	Max Min
✓	deltaPosX	25	Maximize
✓	overallQuality	75	Maximize

Templates

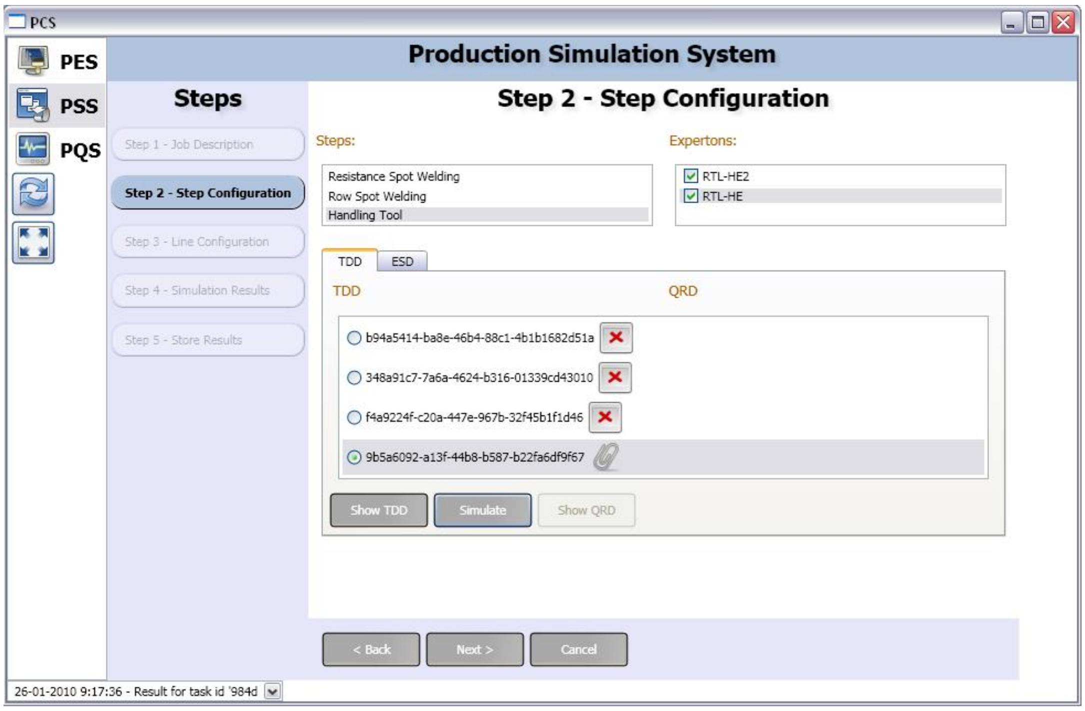
Similar jobs found in the KB:

6494ABB8-D07B-4864-A621-43EBD92F2A92
6494ABB8-D07B-4864-A621-43EBD92F2A92
6494ABB8-D07B-4864-A621-43EBD92F2A92

26-01-2010 9:14:21 - Templates retrieved frc

Figure 6.1: Screenshot of the PSS interface at step 1.

And on the final step, figure 6.6, the list of configurations is ranked by its cost value and the user can select which configurations are worth saving into the knowledge base. After saving, the configuration is available to be ramp into production.



Step 2 - Step Configuration

Steps:
Resistance Spot Welding
Row Spot Welding
Handling Tool

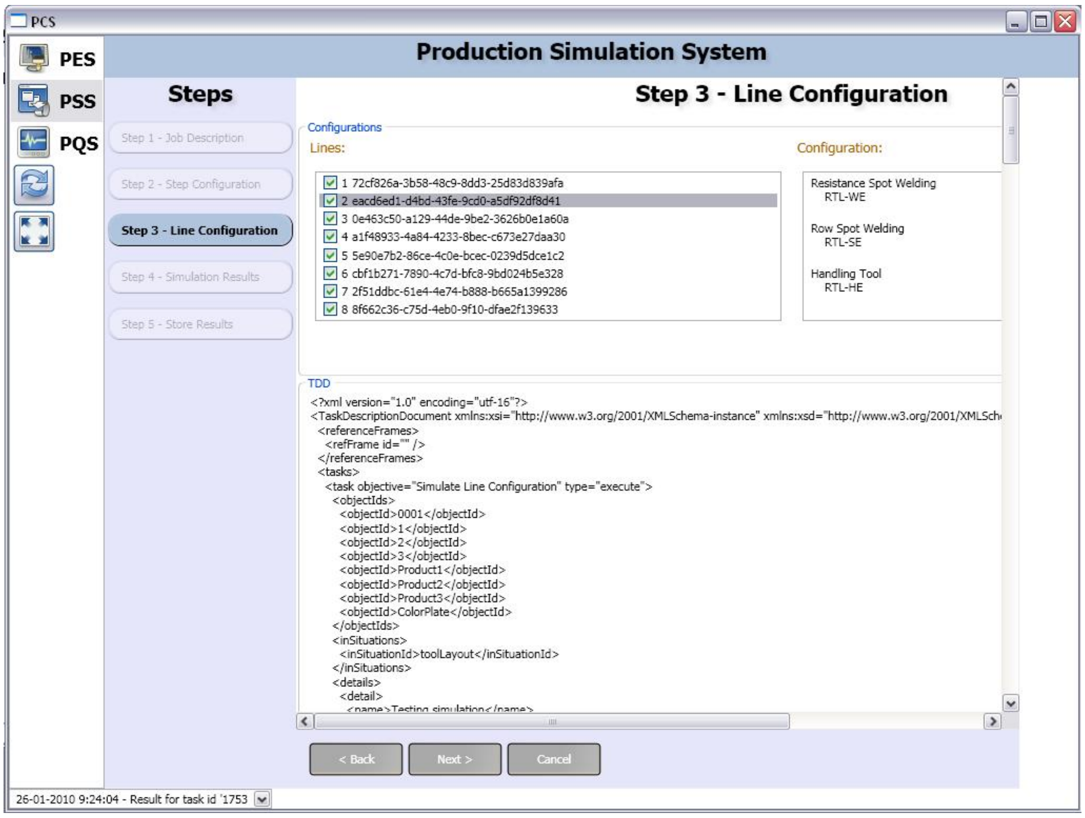
Expertons:
☑ RTL-HE2
☑ RTL-HE

TDD QRD

○ b94a5414-ba8e-46b4-88c1-4b1b1682d51a
○ 348a91c7-7a6a-4624-b316-01339cd43010
○ f4a9224f-c20a-447e-967b-32f45b1f1d46
● 9b5a6092-a13f-44b8-b587-b22fa6df9f67

26-01-2010 9:17:36 - Result for task id '984d

Figure 6.2: Screenshot of the PSS interface at step 2.


PCS
PES
PSS
PQS
Production Simulation System
Steps
Step 1 - Job Description
Step 2 - Step Configuration
Step 3 - Line Configuration
Step 4 - Simulation Results
Step 5 - Store Results
< Back
Next >
Cancel

Step 3 - Line Configuration

Configurations

Lines:
☑ 1 72cf826a-3b58-48c9-8dd3-25d83d839afa
☑ 2 eacd6ed1-d4bd-43fe-9cd0-a5df92df8d41
☑ 3 0e463c50-a129-44de-9be2-3626b0e1a60a
☑ 4 a1f48933-4a84-4233-8bec-c673e27daa30
☑ 5 5e90e7b2-86ce-4c0e-bcec-0239d5dce1c2
☑ 6 cbf1b271-7890-4c7d-bfc8-9bd024b5e328
☑ 7 2f51ddbc-61e4-4e74-b888-b665a1399286
☑ 8 8f662c36-c75d-4eb0-9f10-dfae2f139633

Configuration:
Resistance Spot Welding
RTL-WE

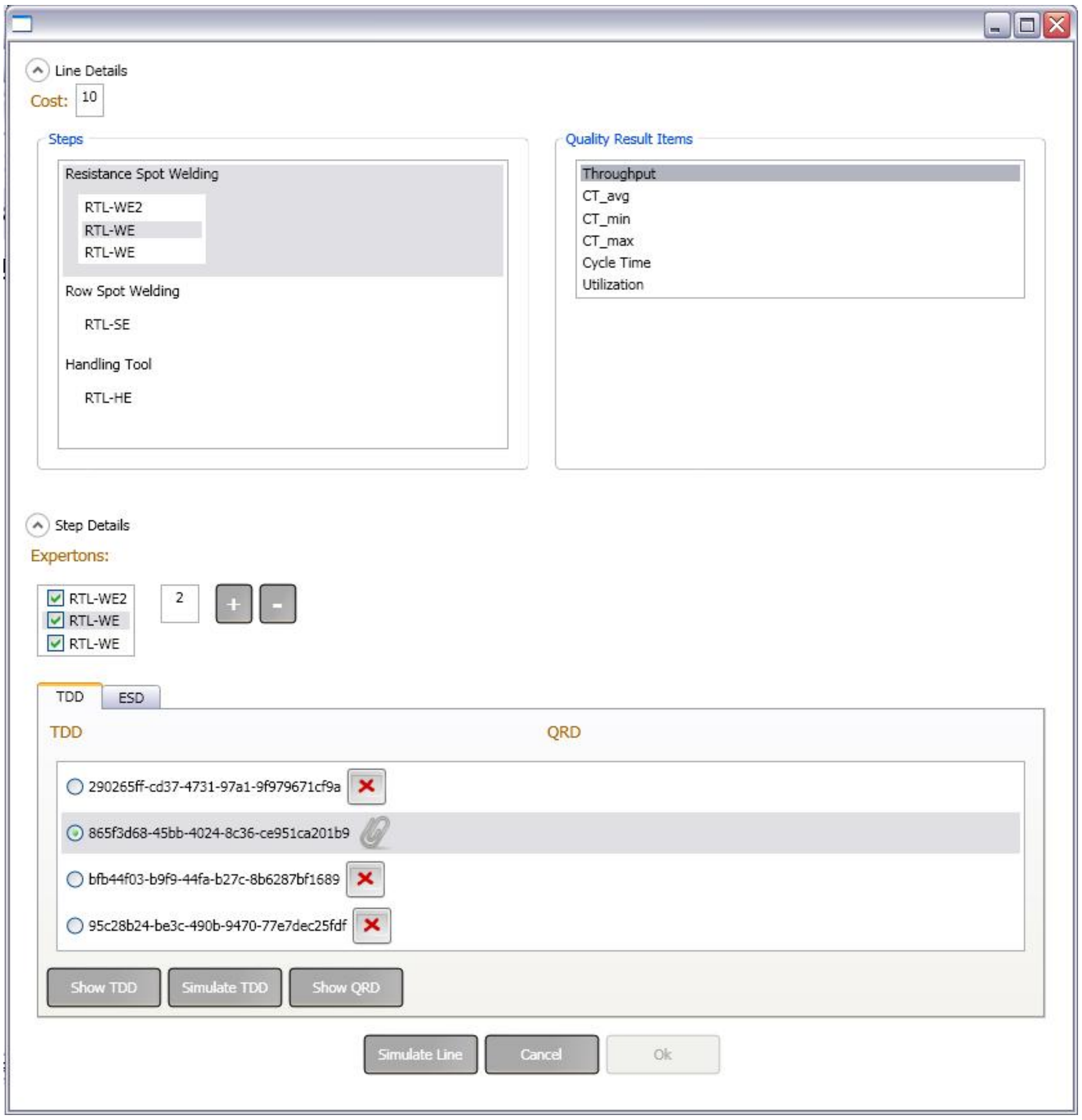
Row Spot Welding
RTL-SE

Handling Tool
RTL-HE

TDD
```
<?xml version="1.0" encoding="utf-16"?>
<TaskDescriptionDocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSch
  <referenceFrames>
    <refFrame id="" />
  </referenceFrames>
  <tasks>
    <task objective="Simulate Line Configuration" type="execute">
      <objectIds>
        <objectId>0001</objectId>
        <objectId>1</objectId>
        <objectId>2</objectId>
        <objectId>3</objectId>
        <objectId>Product1</objectId>
        <objectId>Product2</objectId>
        <objectId>Product3</objectId>
        <objectId>ColorPlate</objectId>
      </objectIds>
      <inSituations>
        <inSituationId>toolLayout</inSituationId>
      </inSituations>
      <details>
        <detail>
          <name>Testing simulation</name>
```

26-01-2010 9:24:04 - Result for task id '1753

Figure 6.3: Screenshot of the PSS interface at step 3.



Cost: 10

Resistance Spot Welding
- RTL-WE2
- RTL-WE
- RTL-WE

Row Spot Welding
- RTL-SE

Handling Tool
- RTL-HE

Throughput
CT_avg
CT_min
CT_max
Cycle Time
Utilization

Expertons:

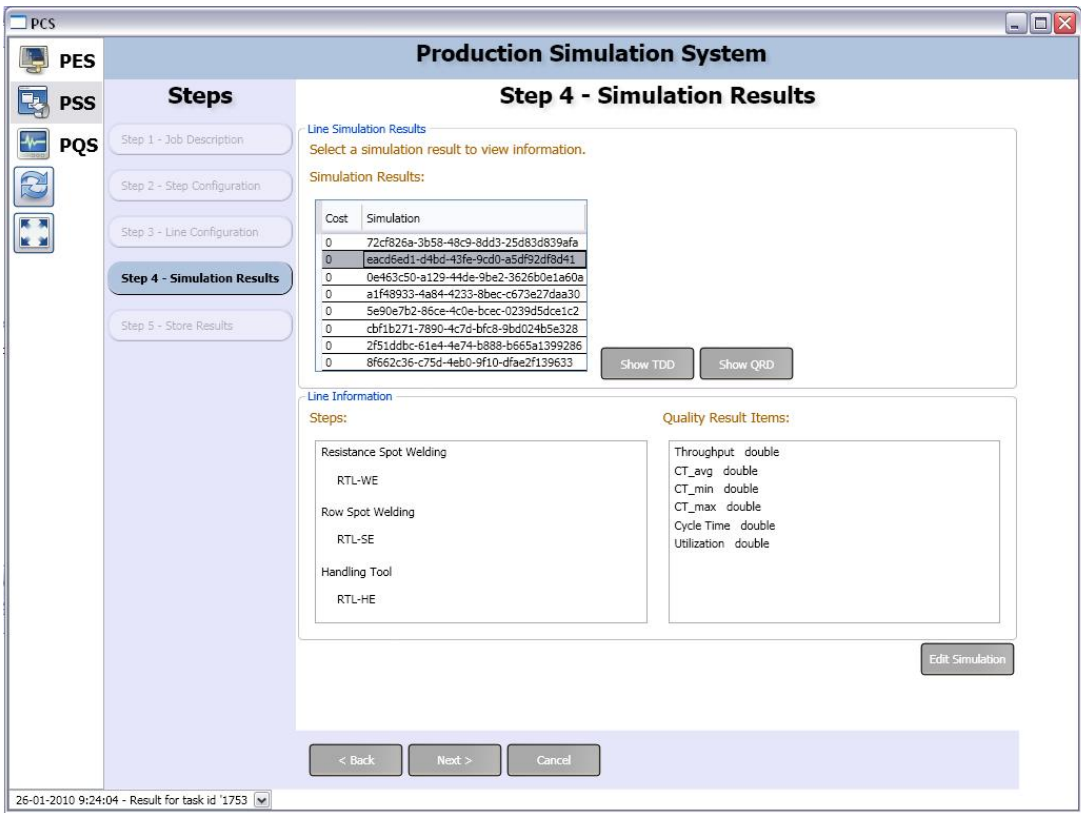
- [✓] RTL-WE2
- [✓] RTL-WE
- [✓] RTL-WE

2

TDD

- ◯ 290265ff-cd37-4731-97a1-9f979671cf9a
- ◉ 865f3d68-45bb-4024-8c36-ce951ca201b9
- ◯ bfb44f03-b9f9-44fa-b27c-8b6287bf1689
- ◯ 95c28b24-be3c-490b-9470-77e7dec25fdf

Figure 6.4: Screenshot of the PSS interface at details of step 3.



Production Simulation System

Step 4 - Simulation Results

Line Simulation Results

Select a simulation result to view information.

Simulation Results:

Cost	Simulation
0	72cf826a-3b58-48c9-8dd3-25d83d839afa
0	eacd6ed1-d4bd-43fe-9cd0-a5df92df8d41
0	0e463c50-a129-44de-9be2-3626b0e1a60a
0	a1f48933-4a84-4233-8bec-c673e27daa30
0	5e90e7b2-86ce-4c0e-bcec-0239d5dce1c2
0	cbf1b271-7890-4c7d-bfc8-9bd024b5e328
0	2f51ddbc-61e4-4e74-b888-b665a1399286
0	8f662c36-c75d-4eb0-9f10-dfae2f139633

Line Information

Steps:

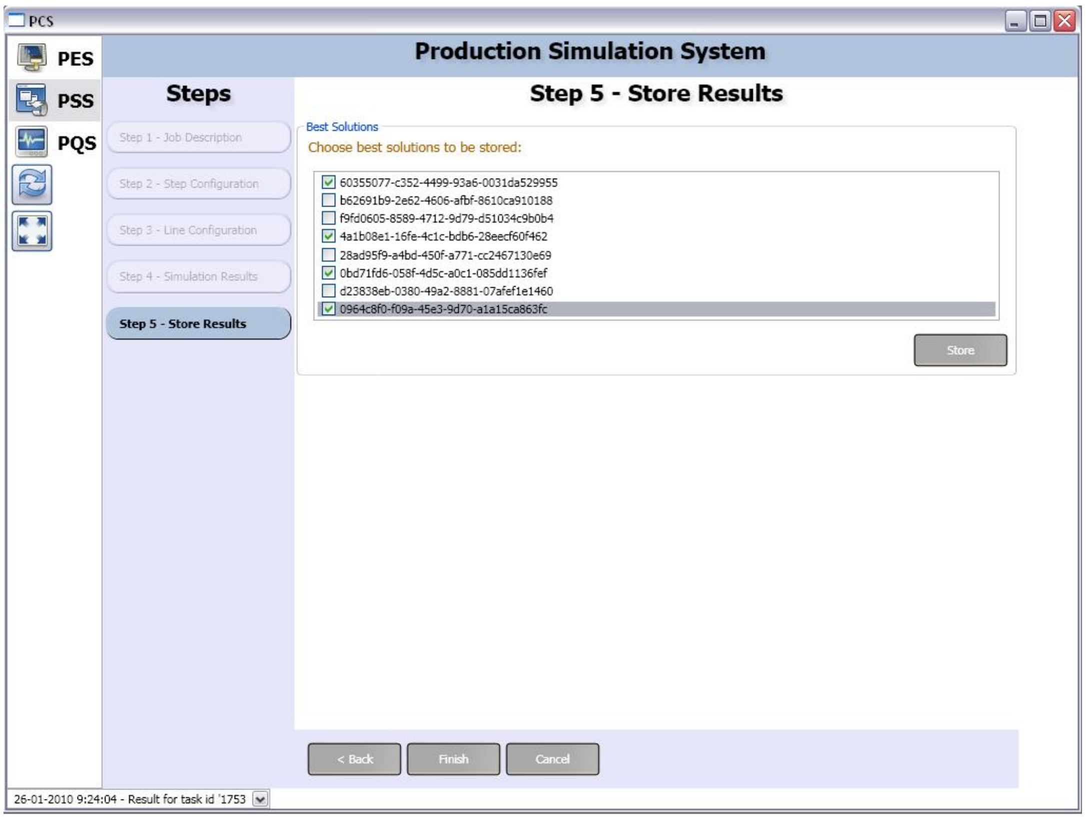
Resistance Spot Welding
RTL-WE
Row Spot Welding
RTL-SE
Handling Tool
RTL-HE

Quality Result Items:

Throughput	double
CT_avg	double
CT_min	double
CT_max	double
Cycle Time	double
Utilization	double

26-01-2010 9:24:04 - Result for task id '1753

Figure 6.5: Screenshot of the PSS interface at step 4.


PCS
PES
PSS
PQS
Production Simulation System
Steps
Step 1 - Job Description
Step 2 - Step Configuration
Step 3 - Line Configuration
Step 4 - Simulation Results
Step 5 - Store Results
Store
< Back
Finish
Cancel

Step 5 - Store Results

Best Solutions

Choose best solutions to be stored:

- ✓ 60355077-c352-4499-93a6-0031da529955
- ☐ b62691b9-2e62-4606-afbf-8610ca910188
- ☐ f9fd0605-8589-4712-9d79-d51034c9b0b4
- ✓ 4a1b08e1-16fe-4c1c-bdb6-28eecf60f462
- ☐ 28ad95f9-a4bd-450f-a771-cc2467130e69
- ✓ 0bd71fd6-058f-4d5c-a0c1-085dd1136fef
- ☐ d23838eb-0380-49a2-8881-07afef1e1460
- ✓ 0964c8f0-f09a-45e3-9d70-a1a15ca863fc

26-01-2010 9:24:04 - Result for task id '1753

Figure 6.6: Screenshot of the PSS interface at step 5.

6.2 Functional Sample

The *XPRESS* project is developing and building its functional sample No 4. The aim of function sample is to provide a demonstrator of the technologies developed under *XPRESS*.

On this particular case, the functional sample No 4, will be the first demonstrator which will include the simulation mechanisms on all expertons, and ultimately, the simulation process coordinated at the PSS level.

Therefore, the functional sample would be the perfect environment to test the integration of the functions developed on this work with real data.

Unfortunately, the development of the functional sample is still on its earlier phase, and will not be subject of this report.

Chapter 7

Conclusions

This chapter concludes the thesis. On it is contained a brief review over the developed work and the results it achieved. At the end of this chapter some suggestions for further research are exposed.

7.1 Review, Discussion and Implications

On this thesis, which started by discussing the industrial manufacturing's need for innovation cycles induced by demanding costumers and a ferocious global competition, presented the european investigation project *XPRESS* lead by a consortium constituted by a group of industries, in conjunction with information and communication technology providers, and academic and research institutions.

The *XPRESS* project has the goal of establishing a breakthrough for the factory of the future with a new flexible production concept based on specialised intelligent process units, called Expertons, integrating a complete process chain.

The main goal of the work is to provide methods to assist the engineers' decision regarding the choice of *Expertons* to use for a specific job description, describing a new product or variant.

Another aim of this research is to provide a comparative study of several approaches that can be taken on this process and their main characteristics.

The output of this work is library containing a set of methods performing the optimization and the comparative evaluation of solutions. Due to the combinatorial explosion effect of the space solution regarding this kind of problems, the approach that was taken lead to a three stage process: ReducedMatrix method; ConfigurationGenerator method; and ConfigurationRanking method. Each stage is characterized by different requirements for the algorithms it could compose.

For ReducedMatrix method was proposed an approach based on the cross-reference efficiency measure of each *Experton* candidate in conjuction with a multicriteria decision process. The ConfigurationGenerator method, on the other hand, was based on a tree data structure with an efficient creation algorithms - the A Star algorithm. Although the ConfigurationRanking method deals with a different kind of data, the mathematical concept is the same as in the ReducedMatrix method. Therefore, the same approach was used as well.

The results published on this thesis show that the optimization strategies proposed can provide viable solutions without penalizing the run time of the software. The results also showed that, for problems with less than two hundred expertons distributed along less than ten tasks, the preemptive effect of the A Star algorithm produces a more significant impact on the quality of the results than on the timesaving it provided. For problems of this size is then prefed to use the analogue non preemptive version, the Best First algorithm. For problems with a higher number of variables, the limitation of the computers memory will, for sure, represent a handicap for the Best First algorithm. This case, there is no other option than using the A Star alternative and accept the risk of not achiving the optimal solutions.

The lack of real data regarding the KPI values, available at a typical production site, implies that the values of the parameters which control these optimizations, may not be suitable for every kind of scenario. In fact, the oposite may also be true, for some scenarios these parameters can also be conservative.

7.2 Main results and Conclusion of the Thesis

The results gathered on this work show the compromises that have to be taken into account whenever there is a decision in stake. The fact of dealing with decisions regarding combinatorial problems introduces, inevitably, the risk of not getting the optimal, or even near the optimal, solution.

Based on the results obtained, the algorithms presented here provide reliable solutions on a relatively short time.

However, this software, does not substitute the planning work of a process engineer at the shop floor of an industy. The restrictions, imposed by the technology under developement at the *XPRESS* project, implies the usage of estimations or simplifications of the data available. Furthermore, as explained on chapter "System Analysis and Design" (chapter 3), there are degrees of freedom which are not taken into consideration, such us the layout, thus the space solution explored with this tool is only a fraction of space solution presented by the problem itself. It is, for sure, an interesting tool to provide comparative data, helping then, on the decision that has to be taken.

7.3 Suggestions for Further Research

Further developments can be done into this work. The results which will be obtained during functional sample will be an essencial element for any improvement on this work. The implementation of a method which would characterize the solution state space of the problem presented, and it would preemptively adjust the optimization parameters in order to obtain the best feasiable results, is also one example of a possible study.

Another area of study can be the analysis of the KPIs available through the experton networked factory and then propose a handcrafted solution for the several multicriteria decisions available.

References

[1] Xpress. Description of work, 2008.

[2] Srinivas Talluri, R.C. Baker, and Joseph Sarkis. A framework for designing efficient value chain networks. *International Journal of Production Economics*, 1999.

[3] Shigeo Shingo. *A Revolution in Manufacturing: The SMED System*. Productivity Press, 1985.

[4] Jim Peterson and Roland Smith. *5S Pocket Guide*. Productivity Press, 1998.

[5] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operational Research*. McGraw-Hill, Sixth edition, 1995.

[6] Wayne L. Winston. *Operational Research - Applications and Algorithms*. Duxbery Press, Third edition, 1993.

[7] A. Charnes, W.W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operational Research*, 1978.

[8] Yannis A. Hajidimitriou and Andreas C. Georgiou. A goal programming model for partner selection decisions in international joint ventures. *European Journal of Operational Research*, 2001.

[9] Armen Zakarian and Andrew Kusiak. Forming teams: an analytical approach. *IIE Transactions*, 1999.

[10] Matthew E. Gaston and Marie desJadins. Agent-organized networks for dynamic team formation. *AAMAS*, 2005.

[11] Tzu-Liang Tseng, Chun-Che Huang, How-Wei Chu, and Rojer R. Gung. Novel approach to multi-functional project team formation. *International Journal of Project Management*, 2004.

[12] Huang Hexin and Chen Jian. A partner selection method based on risc evaluation in virtual enterprises. *International Conference on Services System and Services Management*, 2005.

[13] Gil Tidhar, Anand S. Rao, and Elizabeth A. Sonenberg. Guided team selection. *Proceedings of the Second International Conference on Multi-Agent Systems*, 1995.

[14] Reiko Heckel. Graph transformation in a nutshell. *Dagstuhl Seminar Proceedings*, 2005.

[15] J. Añez, T. De La Barra, and B. Pérez. Dual graph representation of transport networks. *Transportation Research Part B*, 1996.

[16] Eric Klavins, Robert Ghrist, and David Lipsky. Graph grammars for self assembling robotic systems. *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, 2004.

[17] Martin Fowler. *UML Destilled*. Addison Wesley, Second edition, 2000.

[18] Joseph Albahari and Ben Albahari. *C# 3.0 in a Nutshell*. O'Reilly, Third edition, 2007.

[19] GLPK. Glpk - gnu linear programming kit. http://www.gnu.org/software/glpk last visited on january 2010.

[20] Mark A. Weiss. *Data Structures & Algorithms analysis in C++*. Addison Wesley, Second edition, 1999.

Appendix A

Classes Documentation

A.1 EOLibrary.opt_wrapper Class

The **opt_wrapper** class provides the interface to outside the library.

Static Public Member Functions

- static int[ ][ ] **first_opt** (int[ ][ ] matrix, float[ ][ ][ ] KPIs, int[ ] costfunction)

  The first_opt Method reduces the space solution by removing the worse expertons at each task.

  Parameters: ← ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The value is the experton id.

  ← ***KPIs*** is a 3 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The third dimension is indexed to KPIs. The value is the KPI measure.

  ← ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

  Returns: A 2 dimensional array. Is a simplified version of matrix with possibly less expertons per task. The number of expertons reduced is a logarithmical function of the original size.

- static void **second_opt** (int[ ][ ] matrix, float[ ][ ][ ] KPIs, int[ ] costfunction, out int[ ][ ] confs, out float[ ] funcaovalores)

  The second_opt Method computes the n possible best configurations based on the available expertons.

Parameters: $\leftarrow$ ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The value is the experton id

$\leftarrow$ ***KPIs*** is a 3 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The third dimension is indexed to KPIs. The value is the KPI measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

$\rightarrow$ ***confs*** is a 2 dimensional array. The first dimension is indexed to configurations. The second dimension is indexed to steps. The value is the experton id.

$\rightarrow$ ***funcaovalores*** is an array. The array is indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].

- static float[ ] **third_opt** (float[ ][ ] confs, int[ ] costfunction)

  The third_opt Method computes cost function for each configuration.

Parameters: $\leftarrow$ ***confs*** is a 2 dimensional array. The first dimension is indexed to configurations. The second dimension is indexed to KPIs. The value is the KPI measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to KPIs. The value is the KPI weight.

Returns: A array. The array is indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].

A.2 EOLibrary.optmatrix Class

The optmatrix class provides a method for reducing the complexity of the space solution by reducing the number of expertons per task. The final experton quantity per task = 1 + 5.3 $*$ Log10(inicial experton quantity);

Public Member Functions

- int[ ][ ] **get_matrix** ()

  Method for retrieving the new matrix.

Returns: A 2 dimensional array. The first dimension represents tasks. The second dimension represents experton. The value represents the experton id.

- **optmatrix** (int[ ][ ] matrix, float[ ][ ][ ] kpis, int[ ] costfunction)

  Constructor which includes the complexity reduction. The final experton quantity per task = 1 + 5.3 $*$ Log10(inicial experton quantity);

Parameters: ← ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The value is the experton id.

← ***kpis*** is a 3 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The third dimension is indexed to kpis. The value is the kpi measure.

← ***costfunction*** is an array argument. The array is indexed to kpis. The value is the kpi weight.

A.3 EOLibrary.calcCF Class

The **calcCF** class provides a method for computing a cost function of several kpis based on a definition.

Static Public Member Functions

- static float **calc** (float[ ] kpis, int[ ] definition)

  Method for computing a cost function of several kpis based on a definition.

Parameters: ← ***kpis*** is an array with all the kpi measures.

← ***definition*** is an array with the weights of the kpis towards the cost function.

Returns: Result value of the cost funtion.

A.4 EOLibrary.tree Class

The tree class implements a object representing a complete tree.

Public Member Functions

- void **alg_Astar_insertsorted** (int[ ][ ] matrix, float[ ][ ][ ] kpis, int[ ] costfunction, int solutions_-counter_max, int depth, int size_of_queue, int size_of_deletion)

  The alg_Astar_insertsorted Method implements the algorithm for tree creation.

Parameters: ← ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The value is the experton id

← ***kpis*** is a 3 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The third dimension is indexed to kpis. The value is the kpi measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to kpis. The value is the kpi weight.

$\leftarrow$ ***depth*** represents the size of the prediction for calculating the fullpathCF.

$\leftarrow$ ***solutions_counter_max*** represents the maximum number of configurations to be generated.

$\leftarrow$ ***size_of_queue*** represents the maximum size of the node priority queue.

$\leftarrow$ ***size_of_deletion*** represents the quantity of elements of the node priority queue to be eliminated, when the queue reaches it's maximum size.

- **tree** ()

  Constructor for an empty tree.

- **tree** (int[ ][ ] matrix, float[ ][ ][ ] kpis, int[ ] costfunction, int depth, int solutions, int size_of_-queue, int size_of_deletion)

  Constructor with seven arguments.

Parameters: $\leftarrow$ ***matrix*** is a 2 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The value is the experton id

$\leftarrow$ ***kpis*** is a 3 dimensional array argument. The first dimension is indexed to steps. The second dimension is indexed to expertons. The third dimension is indexed to kpis. The value is the kpi measure.

$\leftarrow$ ***costfunction*** is an array argument. The array is indexed to kpis. The value is the kpi weight.

$\leftarrow$ ***depth*** represents the size of the prediction for calculating the fullpathCF.

$\leftarrow$ ***solutions*** represents the maximum number of configurations to be generated.

$\leftarrow$ ***size_of_queue*** represents the maximum size of the node priority queue.

$\leftarrow$ ***size_of_deletion*** represents the quantity of elements of the node priority queue to be eliminated, when the queue reaches it's maximum size.

- void **get_solutions** (out int[ ][ ] results, out float[ ] values)

  Method for retrieving the configurations found at the tree.

Parameters: $\rightarrow$ ***results*** is a 2 dimensional array. The first dimension is indexed to configurations. The second dimension is indexed to steps. The value is the experton id.

$\rightarrow$ ***values*** is an array. The array is indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].

A.5 EOLibrary.nodeTree Class

The **nodeTree** class is the basic element for a tree construction.

Public Member Functions

- **nodeTree** (**nodeData** n)

  Constructor with one argument.

  Parameters: $\leftarrow$ ***n*** **nodeData** for data variable inicialization.

- **nodeTree** (**nodeData** n, **nodeTree** p)

  Constructor with two arguments.

  Parameters: $\leftarrow$ ***n*** **nodeData** for data variable inicialization.
  $\leftarrow$ ***p*** **nodeTree** for parent variable inicialization.

- void **cutdown** ()

  Method that eliminates this. node and all it childs.

Public Attributes

- int **level**

  Variable indicating the depth distance to the root.

- **nodeData data**

  nodeData *element of the node.*

- **nodeTree parent**

  Reference to the parent node.

- **nodeTree previous**

  Reference to the previous node belonging to the same parent node.

- **nodeTree next**

  Reference to the next node belonging to the same parent node.

- **nodeTree child**

  Reference to the first child node.

A.6 EOLibrary.nodeData Class

The **nodeData** class aggregates all the information stored at a **nodeTree**.

Public Member Functions

- **nodeData** (int i, int e, int t, float k, float p, float f, bool r, bool l)

  Constructor with 8 arguments for inicialization of all variables.

  Parameters: $\leftarrow$ ***i*** experton position at matrix.

  $\leftarrow$ ***e*** experton id.

  $\leftarrow$ ***t*** task id.

  $\leftarrow$ ***k*** nodeCF value.

  $\leftarrow$ ***p*** pathCF value.

  $\leftarrow$ ***f*** fullpathCF value.

  $\leftarrow$ ***r*** root flag.

  $\leftarrow$ ***l*** leaf flag.

- **nodeData** ()

  Constructor without inicialization of variables.

- int[ ] **indexs** ()

  Method for retrieving the task id and the experton id of a node.

  Returns: A array with two elements. First element is the task id. Second element is the experton id.

Public Attributes

- int **indexMatrix**

  Experton position at matrix.

- int **nodeExperton**

  Experton id variable.

- int **nodeTask**

  Task id variable.

- float **nodeCF**

  Variable with the computed KPI for node itself.

- float **pathCF**

  Variable with the sum of the computed kpis from the root until the node itself plus the nodeCF.

- float **fullpathCF**

Variable with the sum of pathCF with the minimal computed kpi for the next n steps. The depth n is assigned at the second_opt Method.

- bool **leaf**

  Flag for tree leaf indication.

- bool **root**

  Flag for tree root indication.

A.7 EOLibrary.computeCF Class

The **computeCF** class provides a method for computing the cost function for all configurations.

Static Public Member Functions

- static float[ ] **computeConfs** (float[ ][ ] confs, int[ ] costfunction)

  Method for computing the cost function for all configurations based on a definition.

Parameters: ← ***confs*** is an array with all the kpi measures for the configurations.

← ***costfunction*** is an array with the weights of the kpis towards the cost function.

Returns: A array. The array is indexed to configurations. The value at the position[n] is the computed cost function for the configuration[n].