Internship Report

SoS Optimization: State-of-the-art, Gap Analysis and Optimization Approach

Author:
Meryem Zarrok

Supervised by: Imad Sanduka
University tutor: Prof. Dr. Jean-Stéphane Dhersin

Munich, May-September 2013

Acknowledgments

I would like to thank my supervisor, Mr. Sanduka for the valuable advice and support he has given me throughout this internship. I would also like to thank Mr. Lochow for offering me the opportunity to work within his team.

I take this opportunity to record my sincere thanks to Mr. Lafitte and Mr. Dhersin for their help qnd encouragement. I also thank Mr. Khan for his unceasing encouragement and support.

I am obliged to staff members of EADS, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Contents

Part I
Introduction

1 Motivation

A System-of-Systems (SoS) is a system that joins capabilities of different systems in order to achieve missions, services and goals not achievable by a monolithiic system. SoS appears in many application such us defence application, medical and healthcare management, robotics and sensors, air vehicles, space applications and many other applications. There are many types of SoS (see [4]). This project deals with aknowledged SoS. In this SoS, systems retain their own managment and exhibit many treats such as evolutionary development, geographic distribution, and a wide level of operational independence.

From these characteristics arise many challenges. Indeed, the SoS needs a constant revison of its constituents as well as integeration issues in the case where other systems have to be included. It has also to deal with purpose conflicts - global goals and local goals- of the SoS and its constituents. Since the SoS is large scaled and its constituent could be geographically distributed, it is difficult to maintain a good communication within the SoS. These challenges and many others evolve the complexity in SoS designing and leads to difficulties in predecting emergent behaviours. Hence optimization techniques are needed in order to find the most convenient SoS that fulfil all the requirements.

SoS should enable the decision-makers to understand the implications of various choices on technical performance, costs, extensibility and flexibility over time; thus, effective SoS methodology should prepare decision-makers to design informed architectural solutions for System-of-Systems problems [1]. The design and development of a an SoS go by two main levels: static and dynamic levels. The first level copes with investigations in order to conceptualize the product (SoS). These investigations must take into account many parameters such us customer and stakeholders requirements. While the conception of the SoS, optimization techniques are applied in order to optimize the SoS architechture and its constituents types. In the second level, simulation and analysis are made in order to evaluate the good fonctionning of the product (SoS). This work is made according to a platform which has the abiliy to detect and define SoS failures then reconfigurate it so that it can purssue its tasks without any perturbations. The reconfiguration consists on finding an optimized solution to the probem under the current configuration environment parameters and SoS constituents.

Hence, the aim of the internship is to look for many optimization methods that could be fit in the SoS and develop an optimization engine, which -depending on its intputs- has the ablility to give an optimized solution regarding the case study .

2 Report constituents

The report is organized as folloing: In the first part, we give an overview about the subject studied as well as the company EADS where the internship is taken place. In the second part, we present the optimization methods found in the literature besides their resolution algorithms. The third part copes with the developing of the optimization engine. Finally, in the fourth part, we give a conclusion about the internship.

3 Global view about EADS

The European Aeronautic Defense and Space group (EADS) is a global leader in aerospace, defence and related services. The group was formed in July 10, 2000. It is the fruit of the merger of the German group DaimlerChrylser Aerospace AG (DASA), the French group Aerospatiale-Matra and the Spanich company Construcciones Aeronauticas SA (CASA). The headquarters of the company are located in Toulouse (France) and Leiden (Netherlands). The EADS Group is employing around 133 115 people at more than 170 sites worldwide throught its fourth following companies [2]:

- **Airbus** is the leading commercial and military transport aircraft manufacturer. It employs around 69,300 people at sixteen sites in four European Union countries: France, Germany, the United Kingdom and Spain. In 2011, its capital reached €33.1 billion (incl. €2.5 billion for Airbus Military).

- **Astrium** Astrium provides civil and military space systems and services. It operates via three branches: Astrium Space Transportation, Astrium Satellites and Astrium Services. In 2011, Astrium had a turnover of €5 billion and 16,623 employees in France, Germany, the United Kingdom, Spain and the Netherlands. Astrium is member of Institute of Space, its Applications and Technologies.

- **Eurocopter** is a global helicopter manufacturing and support company. It is the largest in the industry in terms of revenues and turbine helicopter deliveries. In 2011, its turnover is about €5.4 billion with 20,759 employees and more than 11,470 helicopters in service.

- **Cassidian** is the defence and security division of the EADS group and a major provider of global security solutions, lead system integration and aerial, land, naval and joint systems. It is the second largest division of EADS. Until 17 September 2010 it was known as EADS Defence and Security. In 2011, Cassidian has a turnover of €5.8 billion with 20,923 employees and more tham 200 secure networks delivered.

3.1 EADS Innovation Works

Innovation works is a part of EADS group that concerns scientific research and development in a multi-national environment and diverse research fields. It is looking for technology innovations and adaptation, and processes improvements and developments that enhances other EADS group partners (Airbus, Eurocopter, Cassedian, Astrium) in the fields of aerospace, defence, and space industry.

3.2 System engineering Process and Platform team

System engineering Process and Platform team is one of systems engineering and architecture department teams. Its members are distributed over different Europe countries (Germany, Britain, France) with diverse scientific background that related to system engineering process. The team is taking part of different internal (within EADS group) and external (Out of EADS) projects that covers a wide range of system engineering process and development. It provides research and development services in the fields of system engineering methodologies, techniques, processes, and tools.

4 Systems-of-Systems: SoS

Sytems-of-Systems (SoS) are systems-of-interest whose system elements are themselves systems; typically these entail large-scale inter-disciplinary problems involving multiple, heterogeneous, distributed systems. These interoperating collections of component systems usually produce results unachievable by the individual systems alone[3].

Using a SoS approach promotes a new way of thinking for solving unprecedented and complex challenges where the interactions of technology, policy and economics influence the system's development and operation. The new thinking must recognize that, unlike traditional system development, a SoS will likely experience greater influence from policy and economics than technology.

4.1 SoS characteristics

Mark W. Maier has stated in [4] five traits for identifying system of systems challenges. These traits are known as Maier's criteria and are like following:

1. **Operational Independence of the Elements:** If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. The system-of-systems is composed of systems which are independent and useful in their own right.
2. **Managerial Independence of the Elements:** The component systems not only canoperate independently, they dooperate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems.
3. **Evolutionary Development:** The system-of-systems does not appear fully formed. Its development and existence is evolutionary with functions and purposes added, removed, and modified with experience.
4. **Emergent Behavior:** The system performs functions and carries out purposes that do not reside in any component system. These behaviors are emergent properties of the entire system-of-systems and cannot be localized to any component system. The principal purposes of the systems-of-systems are fulfilled by these behaviors.
5. **Geographic Distribution:** The geographic extent of the component systems is large. Large is a nebulous and relative concept as communication capabilities increase, but at a minimum it means that the components can readily exchange only information and not substantial quantities of mass or energy.

4.2 Classification of SoS

In [4], Mair classified systems-of-systems into 3 categories according to their managerial control criterias:

- **Directed:** The SoS in this case is centrally managed. It is built to fulfill specific purposes. The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose. For example, an integrated air defense network is usually centrally managed to defend a region against enemy systems, although its

component systems may operate independently.

- **Collaborative:** Collaborative SoS reffers to systems-of-systems whose central management organization does not have coercive power to acheive SoS purposes. In this case, the component systems collaborate voluntarily to fulfill the agreed upon central purposes. An example for collaborative SoS is the Internet.

- **Virtual:** Virtual SoS lack a central management authority and has no agreed purpose. In this case, large scale behavior emerges and may be desirable. For example, supply chain management is a virtual SoS.

  There is also another type of SoS called **Acknowledge** SoS. This type of SoS has a defined goals and objective and its own management and resources while the constituent systems keep their own independency regarding ownership, funding and goals ([5]).

4.3 SoS applications

SoS has been investigated predominantly in the defence sector. An example for a such application is the air defenses of modern military forces. An integrated air defense system is composed of a geographically dispersed network of semi-autonomous elements. These include surveillance radars, passive surveillance systems, missile launch batteries, missile tracking and control sites, airborne surveillance and tracking radars, fighter aircraft, and anti-aircraft artillery. All units are tied together by a communications network with command and control applied at local, regional, and national centers.

Another application of SoS is the Command and Control Center (CCC) which is a part of the emergency responde system. It operates and monitors communications, detection, and weapons systems essential for controlling air, ground and naval operations. Duties include maintaining and relaying critical communications between air, naval, and ground forces; implementing emergency plans for natural and wartime disasters and also relaying command center information to high-level military and government decision makers. The following figure illustrates an example of CCC.

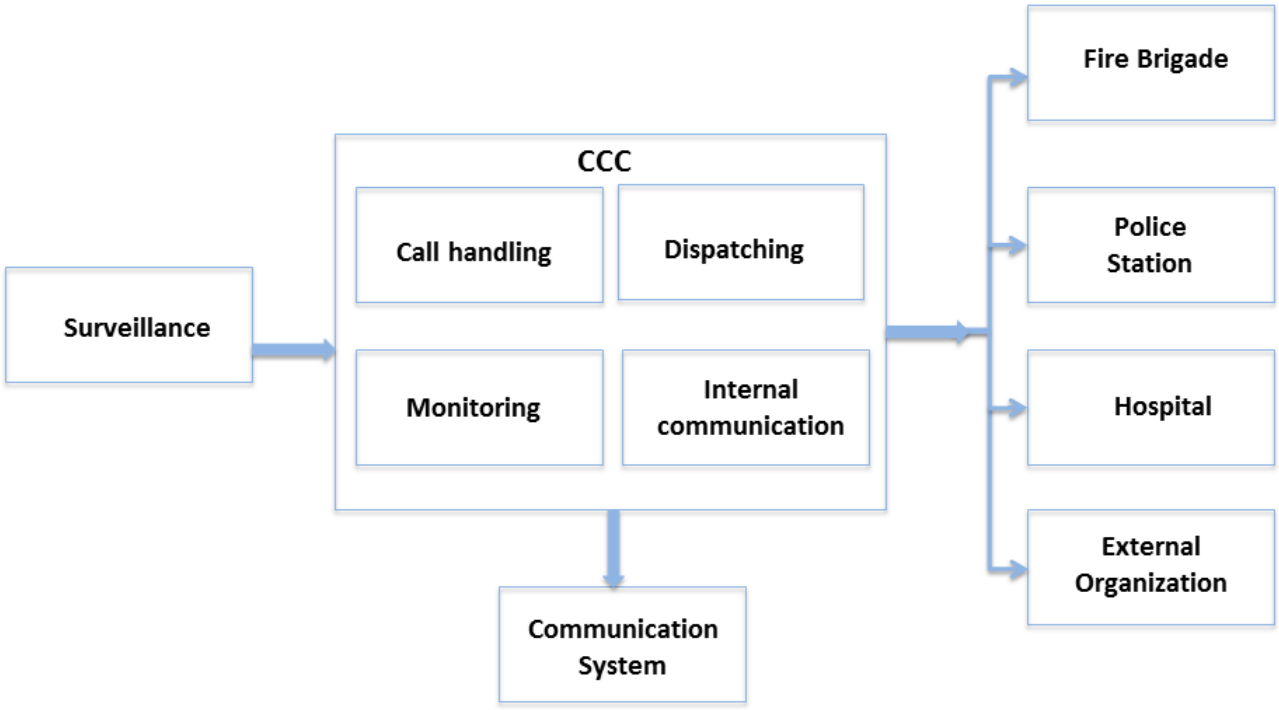

Figure 1: Command and Control Center

This CCC can be used in order to assure an emergency rescue for people detained by a fire in a building. The rescue operation is made as following:

1. CCC receives a call from people who have seen the fire through its call handling systems. The CCC gathers information via the callers and its surveillance systems.

2. Dispatching systems analyse the case then send a dispatching plans to emergency units (fire brigade, police station, hospital or external organization) through communication system.

3. Once the emergency units receive the plans, they handle the emergency operations on the site and report back the site information.

4. The CCC receives the site information and monitors the emergency case.

The internal communication systems assure the communication within the CCC while the communication systems assure the communication between the CCC and the other systems (fire brigade, hospital, police station, hospital or external organization).

Part II
Optimization in SoS

SoS characteristics have many impacts on its operation. In fact, due to the operational independence, the global goal for which the SoS has been built, can conflicts with the local goals of the SoS components. Furthermore, the geographic distirubtion of SoS components make the control of interfaces and communication more complicated, while the evolutionary development can lead to changes in the architecture or SoS operations. Hence, optimization techniques are needed in order to deal with SoS conflicts and to acheive its purposes in the most convenient way.

In the following section, many optimization methods are introduced and classified into three main categories:

1. Optimization in Multi-player environment is used to deal with the multitude of SoS components and to manage systems and SoS purposes.
2. Optimization in Uncertain environment which deals with optimization problems that involve incertainty such as emergent behaviours.
3. Optimization in Dynamic environment is used to deal with problems that exhibit the propreties of overlapping subproblems.

1 Optimization in Multi-player environment

Optimization in Multi-player environment concerns problems where there is various interoperating systems. Each system is autonomous and has the ability to make decisions that affect, at the same time, its operation and SoS operations. Some systems exhibit hierachical traits while decisions are made. This type of problem can be solved with Bi-level optimization. In this case, optimization has to find an equilibrium set of decisions that satisfies all the systems and achieve the SoS purpose. If the problem studied has many goals to achive and has exhibit any hierarchy in decisions, it can be solved via Multi-objective optimization.

1.1 Bi-level Optimization

Bi-level Optimization copes with mathematical programming problems involving two optimization problems called upper (leader) and lower (follower) problems. It is a special case of Multi-level optimization. The bilevel programming problem can be viewed as two-person static Stackelberg game in which control of the decision variables is partitioned among the players who seek to minimize their individual objective functions [6]. Play is sequential and uncooperative. Perfect information is assumed in that both players know the objective functions and allowablee strategies of the other. This type of leader-follower game can be used to model almost any hierarchical system in which two autonomous agents make decisions in a prescribed manner. Applications can be found is such areas as government regulation [7] and market behavior [8]. The general form of Bi-Level Programming Problem (BLPP) is:

$$\begin{aligned} \min_{x} \quad & F(x,y) \\ s.t \quad & G(x,y) \leq 0 \\ & where \;\; y \in argmin \; f(x,y) \\ & s.t \;\; g(x,y) \leq 0 \end{aligned} \tag{1}$$

x and y are vectors of the decision variables of the upper and lower problems, respectively; F and f are the objective functions of the upper and lower problems, while G and g are the upper-level and lower-level constraints. The upper objective is often referred to as the objective of the BLPP, while the lower objective is considred as just a constraint.

In this case, the upper-level problem refers to the SoS and deals with the global goal, while the lower-level problem refers to SoS components and copes with the local goals. According to the structure of the BLPP, the system behaviour is a part of the SoS constraints.

1.1.1 Farmer example:

A salmom breeder, who is located on the edge of a river, has the possibility to produce two races of salmon. Each with its own costs of breeding. The local authority is allowed to raise taxes on the sale price of the salmon. Its aim is to establish a healthy compromise between the company and the level of pollution in the river, given by the criterion F. This problem contain two decision-makers; the local authority who sets taxes and and the breeder who has the control over his production. Decisions are made in this order: First of all, the local authority fixes the appropriate taxes in order to optimize F. Then, the farmer, depending on taxes, fixes the sale price that he could make the greatest possible benefit f. Hence, the upper-variable x refers to the price of taxes and y to the sale price of salmon.

Note : According to the expression of $F,\ G,\ f,\ g$, the BLPP can be devided into many categories. The following section resumes the most commun BLPP problems found in the literature such as Linear Bi-level Programmin Problem (LBPP) and Quadratiq Bi-level Programming Problem (QBPP).

1.1.2 Linear Bi-level Programming Problem (LBPP)

LBPP is similar to standar Linear Programmin (LP) [9] where the objectives functions and constraints are linear. The formulation of the LBPP is given by:

$$\begin{aligned} &\min_{x^1} \quad F(x^1, x^2) = c^{11}x^1 + c^{12}x^2 \\ &where \quad x^2 \ solves \\ &\quad \min_{x^2} \quad f(x^1, x^2) = c^{21}x^1 + c^{22}x^2 \\ &\quad s.t \quad A^1x^1 + A^2x^2 \leq b, \\ &\quad x^1, x^2 > 0, \end{aligned} \tag{2}$$

where $x^1 = (x_1^1, ..., x_{n_1}^1)'$ is a vector of decision variables controlled by the upper level decision maker, $x^2 = (x_1^2, ..., x_{n_2}^2)'$ is a vector of decision variables controlled by the lower level decision maker, A^1 is $m \times n_1$ matrix of coefficients for the upper level decision variables, A^2 is $m \times n_2$ matrix of coefficients for the lower level decision variables, c^{11} is a $1 \times n_1$ vector of profit coefficients of the upper level decision variables in the upper level onjective function F, c^{12} is a $1 \times n_1$ vector of profit coefficients of the lower level decision variables in the upper level onjective function F, c^{21} is a $1 \times n_1$ vector of profit coefficients of the upper level decision variables in the lower level onjective function f, c^{22} is a $1 \times n_1$ vector of profit coefficients of the lower level decision variables in the lower level onjective function f, and b is an $m \times 1$ vector of resource capacity of the system.

For the convenience of analysis, the following notations are intoduced:

$$S^2 = \{(x^1, x^2) |\ A^1x^1 + A^2x2 \leq b,\ (x^1, x^2) \geq 0\} \tag{3}$$

and

$$S^1 = \psi_f(S^2) = \{(\hat{x}^1, \hat{x}^2) \in S^2 |\ f(\hat{x}^1, \hat{x}^2) = \max\{f(x^1, x^2) : (x^2|\hat{x}^1)\}\} \tag{4}$$

where S^1 represents the feasible region of the upper level desicion maker and S^2 is the feasible region of the lower lever decision maker. The maximization in (4) is only taken over x^2 for a fised $\hat{x}^1$.

The geometric properties of the BLPP are more complex than familiar mathematical programmin problems. The convexity of the problem is very important to define the solution and is difficult to establish in some cases. Although the set of rational reactions S^2 may be nonconvex, it does possess some of the important properties of the convex sets. These features are important to establish the solution of the LBPP. Based on these features, Karwan and Bialas developped an algorithm in [10] for solving the problem (2). He took the case where $S^2 = \{x|\ Ax = A^1x^1 + A^2x^2 = b,\ x \geq 0\}$ is bounded with no degeneracy and where the upper level problem has a unique feasible x^1.

The algorithm is based on an extreme point search procedures. It uses most of the standard tools of the simplex method for bounded variables. Furthermore, the solution obtained is a local optimal solution. On rare occasions, it can identify a global optimal solution. The algorithm is as following:

- **Step 1.** Solve the following problem via the simplex method:

$$max\ c^1x \quad s.t \quad Ax = b,\ \ x \geq 0, \tag{5}$$

  where $c^1 = (c^{11}, c^{12})$, $A = (A^1, A^2)$ and $\hat{x} = (\hat{x}^1, \hat{x}^2)$.

- **Step 2.** Set $x^1 = \hat{x}^1$, and solve the following problem via bounded simplex ($l = u = \hat{x}^1$), beginning with basic feasible solution $(\hat{x}^1, \hat{x}^2)$:

$$max\ c^2x \quad s.t \quad Ax = b,\ \ x^1 = \hat{x}^1\ \ x^2 \geq 0. \tag{6}$$

  Let the optimal solution be $\bar{x}$. If $\bar{x} = \hat{x}$, stop; $\hat{x}$ is a global optimal solution. Otherwise, go to Step 3a with current basis $\bar{B}$ and relax the constraints $x^1 = \hat{x}^1$.

- **Step 3a.** If all nonbasic variables are equal to zero, go to step 4 with curent basis $\bar{B}$. Otherwise, go to Step 3b.

- **Step 3b.** If $\bar{b}_i > 0$ for all i, then go to Step 3c. Otherwise consider $\bar{b}_s = 0$. Choose $y_{s,N}$ such that $1 \leq j \leq k$ and $y_{s,N_j} \neq 0$. Bring $x^1_{N_j}$, into the basis via a degenerate pivot. Go to Step §a.

- **Step 3c.** Consider any nonbasic level one variable which is at a strictly positive value, say $x^1_{N_j}$. If $r_j \leq 0$, then increase the value of $x^1_{N_j}$, bringing it into the basis until another variable is forced out of the basis. If $r_j > 0$, then decrease the value of x^1_N, until it eitheur reaches zero, or forces another variable out of the basis. Go to Step 3a.

- **Step 4.** Beginning with the current basis $\bar{B}$, solve the following problem via a modified simplex procedure:

$$\max\ c^2x \quad s.t: \quad A^2x^2 = (b - A^1\tilde{x}^1), \quad x^1 = \tilde{x}^1,\ \ x^2 \geq 0. \tag{7}$$

If $x^* = \tilde{x}$, then enter the candidate into the basis. Repeat Step4 until no more candidates exist which satisfy the above conditions, then stop. The resulting solution is a local optimal solution to (2).

A complete proof of convergence of the above procedure is provided in [11]. The algorithm begins by finding the maximum of c^1x over the entire feasible region, S^2. A check in Step 2 determines if the resulting solution is also an element of S^1. If it is, the algorithm terminates with a global (not simply local) optimal solution. Otherwise, the algorithm finds a point, $\bar{x}$, in S^1 whose x^1 coordinates are the sameas the solution in Step 1.

The purpose of Step 3 is to relax the constraint $x^1 = \bar{x}^1$, and move to an extreme point, x^*, of S^1 such that $c^1x^* \geq c^1\bar{x}$. The vector $\bar{x}$ is guaranteed to be an element of S^1 simply by its definition. The extreme point x^* must be also in S^1 (see Theorem 5.1 in [10]) since x^* positively contributes in a convex combination which forms $\bar{x}$.

Once the algorithm has found an extreme point in S^1, it then moves among extreme points of S^1, never allowing c^1x to decrease. The algorithm terminates with an extreme point solution in S^1 which has the property that all adjacent extreme points either lead to a decrease in c^1x or do not belong to S^1. Thus a local optimal solution is obtained.

1.1.3 Linear-Quadratic Bi-level Programming Problem (LQBPP)

Here, another case of the BLPP called the "linear/quadratic" bilevel problem, where the upper-level objective function is linear whereas the lower-level objective function is quadratic. The problem is given by:

$$\max_x \; F(x,y) = c^1x + c^2y, \tag{8}$$

$$s.t \qquad x \in X = \{x : Dx \geq d\} \tag{9}$$

$$\max_y \; f(x,y) = c^3y + x^TQ_1y + \frac{1}{2}y^TQ_2y, \tag{10}$$

$$s.t \quad g(x,y) = Ax + By \geq b, \tag{11}$$

$$y \in Y = \{y : Ey \geq e\} \tag{12}$$

where $F : X \times Y \to R^1$ is linear and $f : X \times Y \to R^1$ is quadratic, A is $m_1 \times n_1$, B is $m_2 \times n_2$, E is $m_3 \times n_2$, Q_1 is $n_1 \times n_2$, Q_2 is $n_2 \times n_2$ symmetric, negative semidefinite, and c^1, c^2, c^3, b, d, and e are vectors of conformal dimension.

In [12], Brand and Moore have developed a branch and bound algorithm for solving the problem (8)-(12). The algorithm needs the following notations and assymptions:

Let the follower's (lower problem) Rational Reaction Set be:

$$M(x) = \{y : \; y = argmax[f(x,y) : y \in Y, g(x,y) \geq b]\}. \tag{13}$$

and the Inducible Region:

$$R = \{(x,y) : x \in X, y \in M(x)\} \tag{14}$$

In order to assure that (8)-(12) is well posed, the feasible region (9), (11) and (12) is assumed to be nonempty and compact, and that for each decision taken by the leader, the follower variable

has some room to respond. The rational set $M(x)$ defines this response while the inducible region R represents the set over which the leader may optimize when given control of all the variables. Furthermore, $M(x)$ is assumed to be a point-to-point map. Hence, the problem solved by branch and bound algorithm is

$$\max\{F(x,y) : (x,y) \in R\}. \tag{15}$$

In order to make the problem more manageable, it is reformulated as a standard mathematical program by exploiting the follower's Kuhn-Tucker conditions.

Karush-Kuhn-Tucker theorem necessary conditions : Let x^* be a local minimum of the problem

$$\begin{aligned} \min \ & f(x) \\ s.t \quad & h_1(x) = 0, ..., h_m(x) = 0 \\ & g_1(x) \leq 0, ..., g_r(x) \leq 0 \end{aligned} \tag{16}$$

where f, h_i, g_i are continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$, and assume that x^* is regular. Then there exist unique Lagrance multiplier vectors $\lambda^* = (\lambda_1^*, ..., \lambda_m^*)$, $\mu^* = (\mu_1^*, ..., \mu_r^*)$ such that

$$\begin{aligned} & \nabla_x L(x^*, \lambda^*, \mu^*) = 0, \\ & \mu_j^* \geq 0, \quad j = 1, ..., r, \\ & \mu_j^* = 0, \ \forall j \notin A(x^*) \end{aligned} \tag{17}$$

where $A(x^*)$ is the set of active constraints at x^* and L is the Lagrangien given by

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{r} \mu_i g_i(x)$$

With applying Karush-Kuhun-Tucker conditions to the follower's problem, the problem becomes:

$$\max_x F(x,y) = c^1 x + c^2 y, \tag{18}$$

$$s.t \ \ x^T Q_3 y + y^T Q_2 y + u^1 B + u^2 I = -c^3, \tag{19}$$

$$u^1(Ax + By - b) + u^2 y = 0, \tag{20}$$

$$Ax + By \geq b, \tag{21}$$

$$x, y, u^1, u^2 \geq 0, \tag{22}$$

where u^1 and u^2 are $m-$ and n_2- dimensional Kuhn-Tucker multipliers, and I is an $n_2 \times n_2$ identity matrix. Constraints (19), (20) and (22) can be interpreted as an explicit representation of the inducible region.

As suggested in [12], the basic idea of the algorithm discussed is to suppress the complementarity term (20) and solve the resulting linear program. At each iteration, a check is made to see if (20) is satisfied. If so, the corresponding point is a potential solution to (18) − (22); if not a branch and bound scheme is used to implicitly examine all combinations of complementary slackness.

The following notations are used in the algorithm. Define $u = (u^1, u^2)$, let $W = 1, ..., m + n_2$ be the index set for the complementarity term (20), and let $\underline{F}$ be the incumbent lower bound on the upper-level objective function, $W_k \subseteq W$ the kth level of the branch and bound tree, and P_k (with $|W_k|$ nonzero components) corresponding to an assignment of either $u_i = 0$ or $g_i = 0$ for $i \in W_k$. Define also the following sets:

$$S_k^+ = i : i \in W_k \text{ and } u_i = 0,$$
$$S_k^- = i : i \in W_k \text{ and } g_i = 0,$$
$$S_k^0 = i : i \notin W_k i.$$

Fot $i \in S_k^0$, the variables u_i and g_i are free to assume any nonnegative in the solution of $(18) - (22)$ with (20) omitted, so (20) will not necessarily be satisfied. The algorithm is as follow:

Algorithme 1 Branch and Bound for LQBLPP

Step 0. (Initialization). Put $k = 0$, $S_k^+ = \emptyset$, $S_k^- = \emptyset$, $S_k^0 = \{1, ..., m + n_2\}$, and $\underline{F} = -\infty$.

Step 1. (Iteration k). Set $u_i = 0$ for $i \in S_k^+$ and $g_i = 0$ for $i \in S_k^-$. Attempt to solve (18)-(22) without (20). If the resultant subproblem is infeasible, go to *Step* 5 ; otherwise, put $k \leftarrow k + 1$ and label the solution (x^k, y^k, u^k).

Step 2. (Fathoming). If $F(x^k, y^k) \leq \underline{F}$, go to *Step* 5.

Step 3. (Branching). If $u_i^k \cdot g_i(x^k, y^k) = 0$, $i = 1, \cdots, m + n_2$, go to *Step* 4; otherwise, select i for which $u_i^k \cdot g_i(x^k, y^k)$ is largest and label it i_1. Put $S_k^+ \leftarrow S_k^+ \cup \{i_1\}$, $S_k^0 \leftarrow S_k^0 \setminus \{i_1\}$, $S_k^- \leftarrow S_k^-$, append i_1 to P_k, and go to *Step* 1.

Step 4. (Updating). $\underline{F} = F(x^k, y^k)$.

Step 5. (Backtracking). If no live node exists, go to *Step* 6. Otherwise branch to the newest live vertex and update S_k^+, S_k^-, S_k^0 and P_k as discussed below. Go to *Step* 1.

Step 6. (Termination). If $\underline{F} = -\infty$, there is no feasible solution to $(18) - (22)$. Otherwise, declare the feasible point associated with $\underline{F}$ the optimaö solution.

Step 1 is designed to find a new point that is potentially bilevel feasible. If no solution exists, or the solution does offer an amelioration over Step 2, the algorithm goes to Step 5 and backtracks. At Step 3, a check is made to determine if the complementary slackness conditions are satisfied. In practice, if $|u_i \cdot g_i| < 10^{-6}$, it is considered to be zero. Confirmation indicates that a feasible solution of the bilevel program has been found, and Step 4, the lower bound on the leader's objective function is updated. Alternatively, if the complementary slackness conditions are not satisfied, the term with the largest product is used at Step 3 to provide the branching variable. Branching is always done on the Kuhn-Tucker multiplier.

At Step 5, the backtracking operation is performed. A live node is one associated with a subproblem that has not yet been fathomed at either Step 1 due to infeasibility or at Step 2 due to bounding, and whose solution violates at least one complementary slackness condition. To facilitate bookkeeping, the path P_k in the depth of the tree. Indices only appear in P_k if they are in either S_k^+ or S_k^- with the entries underlined if they are in S_k^-. Because the algorithm always branches on Kuhn-Tucker component of P_k, underlined entry is deleted from S_k^+ and added to S_k^-; the erased entries are deleted from S_k^- to S_k^0.

Once Step 6 is reached and $\underline{F} = -\infty$, it can be conclued that the original constraint region (9), (11), (12) is empty. This will only be the case if the first subproblem at Step 1 is infeasible. Alternatively, the algorithm terminates with the incumbent whose optimality is now established. The following proposition shows that the algorithm terminates.

Proposition 1 *Under the uniqueness assumption associated with the rational reaction set $M(x)$, the algorithm terminates with the global optimum of the LQBLPP* (18) − (22).

The proof is in [12].

Note: This algorithm can be also applied in ordor to solve the linear case of BLPP. Consider the following BLPP:

$$\begin{aligned} &\max_{x^1} \quad F(x^1, x^2) = c^{11}x^1 + c^{12}x^2 \\ &where \quad x^2 \ solves \\ &\quad max_{x^2} \quad f(x^1, x^2) = c^{21}x^1 + c^{22}x^2 \\ &\quad s.t \quad A^1x^1 + A^2x^2 \geq b, \\ &\quad x^1, x^2 > 0. \end{aligned} \tag{23}$$

KKT conditions and the linear Complementarity problem Consider the linear problem

$$\max_x \{cx : Ax \geq b, x \geq 0\} \tag{24}$$

The corresponding Lagrangien is

$$L(x, \lambda, \mu) = cx - \lambda^T(Ax - b) - \mu^T x \tag{25}$$

where λ and μ are the $m-$ and $n-$ dimensional Kuhn-Tucker or Lagrange multipliers for the technological and nonnegativity constraints, respectively. The Kuhn-Tucker necessary conditions for optimality gives:

$$\begin{aligned} &\nabla_x L(x, \lambda, \mu) = c - \lambda^T A - \mu^T = 0 \\ &\nabla_\lambda = Ax - b \geq 0 \\ &\nabla_\mu L(x, \lambda, \mu) = x \geq 0 \\ &\lambda^T(Ax - b) = 0 \\ &\mu^T x = 0 \\ &\lambda, \mu \geq 0. \end{aligned} \tag{26}$$

Hence, the previous algorithm can also be applied in order to solve the linear case of BLPP.

1.1.4 Mixed Integer Two-level Linear Programming Problem

In some hierarchical structure system, the high-level decision making situations often require the inclusion of zero-one variables which present "yes-no" decisions. In [13], Wen and Yang presented a Mixed Integer Two-Level Linear programming problem, where they combine the zero-one decision variables controlled by the high-level decision maker with the real-valued decisions variables controlled by the low-level decision maker. The general form of the problem is given by:

$$
\begin{array}{rll}
(MITLP): & \max & \{f_1(x^1,x^2)=c^{11}x^1+c^{12}x^2:(x^1)\} \\
& st: & \max\{f_2(x^1,x^2)=c^{22}:(x^2|\hat{x}^1)\} \\
& & st:\ A^1x^1+A^2x^2\leq b \\
& & st:\ x^1=(x_1^1,x_2^1,...,x_{n_1}^1), \\
& & x_j^1\in\{0,1\},\ j=1,2,...,n_1, \\
& & x^2\geq 0
\end{array}
$$

where $\max\{f_1(x^1,x^2)=c^{11}x^1+c^{12}x^2:(x^1)\}$ denotes the maximum of f_1 over x^1, x^2 but only x^1 can be controlled at the high level problem; $\max\{f_2(x^1,x^2)=c^{22}x^2:(x^2|\hat{x}^1)\}$ denotes the maximum of f_2 over x^2 for a fixed value of $\hat{x}^1$; $x^1\in\mathbb{R}^{n_1}$, $x^2\in\mathbb{R}^{n_2}$, $A^1\in\mathbb{R}^{m\times n_1}$, $A^2\in\mathbb{R}^{m\times n_2}$, $c^{11}\in\mathbb{R}^{n_1}$, $c^{12}\in\mathbb{R}^{n_2}$, $c^{22}\in\mathbb{R}^{n_1}$ and $b\in\mathbb{R}^m$.

Let $W_{f_2}(S)$ be the feasible region of the high-level decision-maker such as:

$$W_{f_2}(S)=\{(\hat{x}^1,\hat{x}^2)\in S|f_2(\hat{x}^1,\hat{x}^2)=\max\{f_2(x^1,x^2):(x^2|\hat{x}^1)\}\},$$

where

$$S=\{(x^1,x^2)|A^1x^1+A^2x^2\leq b,\ x_j^1\in\{0,1\},\ j=1,2,...,n^1;\ x^2\geq 0\}$$

This model assumes that A^2 and $b\ \geq 0$, S is nonempty and the optimal solution of MITLP is nondegenerate.

Wen and Yang developed an algorithm based on a Branch and Bound method in order to solve the MITLP. The lower bound on the optimal value of the high-level objective function can be determined as the greatest value of the high-level objective function found so far. Hence, the problem is to determine a lower bound. These notations are introduced in order to develop the bounding function:

- k: the order number of generated node in a branch-and-bound tree.
- $J_k^0=\{j|x_j^1$ is free binary variable, $j=1,2,...,n_1\}$.
- $J_k^+=\{j|x_j^1$ is fixed at 1, $j=1,2,...,n_1\}$.
- $J_k^-=\{j|x_j^1$ is fixed at 0, $j=1,2,...,n_1\}$.

For the MITLP, if part of the high-level decision variables are fixed, the problem becomes:

$$
\begin{array}{rll}
(MITLP)_f: & \max & f_1=\sum_{j\in J_k^0}\ c_j^{11}x_j^1+\sum_{j\in J_k^+}\ c_j^{11}+\sum_{j=1}^{n_2}\ c_j^{12}x_j^2 \\
& st: & \max f_2=\sum_{j=1}^{n_2}c^{22} \\
& & st:\sum_{j\in J_k^0}\ a_j^1x_j^1+\sum_{j=1}^{n_2}\ a_j^2x_j^2\leq b-\sum_{j\in J_k^+}a_j^1 \\
& & \ \ x_j^1\in\{0,1\},\ j\in J_k^0, \\
& & x^2\geq 0
\end{array}
$$

where a_j^1 is the jth column vector of A^1, a_j^2 is the jth column vector of A^2.

Furthermore, by neglecting the low-level objective function, f_2 of $MITLP_f$, the resulting problem becomes the following mixed integer linear programming problem :

$$
\begin{array}{lll}
(MILP)_f: & \max & f_1 = \sum_{j\in J_k^0}\ c_j^{11}x_j^1 + \sum_{j\in J_k^+}\ c_j^{11} + \sum_{j=1}^{n_2}\ c_j^{12}x_j^2 \\
 & & st: \sum_{j\in J_k^0}\ a_j^1x_j^1 + \sum_{j=1}^{n_2}\ a_j^2x_j^2 \le b - \sum_{j\in J_k^+} a_j^1 \\
 & & x_j^1 \in \{0,1\},\ j \in J_k^0, \\
 & & x^2 \ge 0
\end{array}
$$

Hence, the high-level optimal objective value of $MITLP_f$ is less than or equal to the optimal objective value of $MILP_f$ (see [13]). The algorithm proposed in [13] to solve this problem is based on the following theorem:

Theorem 2 *Consider the following problem:*

$$
\begin{array}{lll}
(B): & \max\ Z_B = \sum_{j=1}^{n_2} c_j^{12}x_j^2 \\
 & st: \quad \sum_{j=1}^{n2}\ a_j^2x_j^2 \le b \\
 & x_j^2 \ge 0,\ j = 1,2,...,n_2.
\end{array}
$$

Then, if Z_B^* is the optimal objective value of (B) and Y_B^* is the dual optimal solution of (B), then the high-level optimal objective value of $MITLP_f$ is less than or equal to the value Z^u, where

$$
Z^u = Z_B^* + \sum_{j\in J_k^+}(c_j^{11} - Y_B^*a_j^1) + \sum_{j\in J_k^0}\max\{c_j^{11} - Y_B^*a_j^1, 0\},
$$

i.e. Z^u is an upper bound of $(MITLP_f)$.

The proof is in [13].

Based on the previous theorem, he upper bounf of node k in the tree can be found by solving the problem (B). The branching procedure always chose the first free variable to branch upon by first setting the corresponding binary variable to zero and then setting it to one. There are two important reasons for this: one is to reduce the number of performing dual simples pivot operationsm the other is to save the memory storage space. Especially, the first reason will greatly affect the efficiency for solving the MITLP (see [13]). The exact algorithm is described as follows:

- **Step 1a:** (INITIALIZATION) Set $N = 0$, $K = 0$, $J_k^0 = \{1,2,...,n_1\}$, $J_k^+ = J_k^- = \emptyset$ and solve the following problem (F):

$$
\begin{array}{ll}
\max & \sum_{j=1}^{n_2} c_j^{22}x_j^2 \\
st: & \sum_{j=1}^{n_2} a_j^2x_j^2 \le b \\
 & x_j^2 \ge 0,\ j = 1,2,...,n_2
\end{array}
$$

  with the optimal solution x^{2^*} and objective value Z^*.

- **Step 1b.** Solve the problem (B) with the optimal objective value Z_B^* and the dual optimal solution Y_B^*. Calculate $H(j) = c_j^{11} - Y_B^*a_j^1,\ j = 1,2,...,n_1$.

- **Step 2.** (BRANCHING) Set $N = N + 1$, $J_k^0 = J_k^0 \setminus \{N\}$, $J_k^- = J_k^- \cup \{N\}$, $N = N + 1$.

- **Step 3:** (CALCULATING BOUND) $Z^u = Z_B^* + \sum_{j \in J_k^+} H(j) + \sum_{j \in J_k^0} \max\{H(j), 0\}$.

- **Step 4a:** (FATHOMING) If $Z^u \leq Z^*$, go to Step 5a; otherwise, go to Step 4b.

- **Step 4b:** Check if $N = n_1 - 1$, set $N = N + 1$, $J_k^0 = J_k^0 \setminus \{N\}$, $J_k^- = J_k^- \cup \{N\}$, $k = k+$ and go to Step 6a; otherwise, go to Step 2.

- **Step 5a:** (BACKTRACKING) If $N \in J_k^-$, set $J_k^+ = J_k^+ \cup \{N\}$, $J_k^- = J_k^- \setminus \{N\}$, $k = k + 1$ and go to Step 3; otherwise go to Step 5a.

- **Step 6a:** (CALCULATING FEASIBLE SOLUTION) Solve the following problem (L):

$$\begin{aligned} &\max \ \textstyle\sum_{j=1}^{n_2} c_j^{22} x_j^2 \\ &st : \textstyle\sum_{j=1}^{n_2} a_j^2 x_j^2 \leq b - \sum_{j \in J_k^+} a_j^1 \\ &\quad x_j^2 \geq 0, \quad j = 1, 2, ..., n_2 \end{aligned}$$

  Let x^{2L} be the optimal solution of (L) and set

$$Z^L = \sum_{j=1}^{n_2} c_j^{12} x_j^{2L} + \sum_{j \in J_k^+} c_j^{11}.$$

- **Step 6b:** If $Z^L > Z^*$, update $(x^{1*}; x^{2*})$ and Z^* by $(x^{1L}, ;^{2L})$ and Z^L, respectively. Go to Step 6c.

- **Step 6c:** If $N \in J^-$, set $J_k^+ = J_k^+ \cup \{N\}$, $J_k^- = J_k^- \setminus \{N\}$, $k = k + 1$ and go to Step 6a; otherwise, go to Step 5b.

- **Step 7:** (TERMINATION) Stop; $(x^{1*}; x^{2*})$ is the optimal solution with the high-level optimal objective value Z^*.

Step 1a is to find an initial feasible solution of MITLP by setting all the high-level decision variables to zero via the simplex method. Step 1b is designed to find the basic information about bounds. Branching occurs at Step 2 where J_k is updated. Step 3 is to calculate the upper bound. Step 4 is to check if the upper bound is less than or equal to the current optimal objective value. If so, this node may terminate; otherwise, it is necessary to move forward to branch. Backtracking procedure is adopted by branching to the newest live node which takes place in Step 5. To find a potentially better solution for MITLP is accomplished in Step 6. Finally, if there exists no live node, the solution procedure will terminate with the current optimal solution in Step 7.

Note: The branch-and-boung technique is often an effective tool for dealing with the problems in mixed zero-one form. Its efectiveness depends on the design of the bound and the selection of branching rules. In the worst case, when the number of high-level zero-one decision variables grows linearly, the computational times grows exponentially.

1.1.5 Nonlinear Bi-level Programming Problem

Consider the general form of BLPP (1), for the case where the functions F, f, g and h are nonlinear and nonconcex, to date only few algorithms exist for the problem with any degree of success. The existing algorithms may fall into three classes. The first one utilizes the Karush-Kuhn-Tucker (KKT) conditions on the lower-level problem as constraints on the upper level problem. Brad and Edmunds in [14] have developped an algorithm based on branch and bound method in order to solve the nonconvex mathematical problem, which result from the application of the KKt conditions.

The second set of algorithms are based on descent approaches for the upper level problem with subgradient information from the lower level problem aquired in a variety of ways. In this sets of algorithms, the BLPP may be viewed solely in terms of the upper variables. Assuming that, for any x, the optimal solution of the lower-level problem is unique and defines y as an implicit function $y(x)$ of x. Descent methods consist on finding a feasible direction d for a given feasible point x so as to ensure a resonable deacrease of the upper-level objective function F. The major issue is the availability of the gradient (or sub-gradient) of the upper-level objective $\nabla_x F(x, y(x))$, at a feasible point. Applying the chain rule of differentiation, it results, whenever $\nabla_x y(x)$ is well defined:

$$\nabla_x F(x, y(x)) = \nabla_x F(x, y) + \nabla_y F(x, y) \nabla_x y(x) \tag{27}$$

Unfortunately, $y(x)$ may not be everywhere differentiable, in which case the functions of the upper-level problem will not be differentiable everywhere. Furthermore, without significant restrictions on the lower-level problem, the upper-level problem may not be convex. In[15], Lasdon have proposed a method for approximating this gradient. The procedure for computing this gradient requires solving a linear system of size $n+m$, where n is the dimension of y and m the dimension of f.

Gaunvin in [15] shows that an upper-level descent direction at a given point x is a vector $d \in \mathbb{R}^{n_1}$ such that:

$$\nabla_x F(x, y^*) + \nabla_y F(x, y^*) w(x, d) < 0, \tag{28}$$

when $y^* = y(x)$ and $w \in \mathbb{R}^{n_2}$ is a solution of the program

$$\begin{aligned} &\min_w (d^T, w^T) \nabla^2_{xy} L(x, y^*, \lambda)(d, w) \\ &s.t\ \nabla_y g_i(x, y^*) w \leq -\nabla_x g_j(x, y^*) d,\ i \in J, \\ &\nabla_y g_j(x, y^*) w = -\nabla_x g_i(x, y^*) d,\ i \in I(x), \\ &\nabla_y f(x, y^*) w = -\nabla_x f(x, y^*) d + \nabla_x L(x, y^*, \lambda) d \end{aligned} \tag{29}$$

with

$$I(x) = \{i \in I : g_i(x, y^*) = 0\} \tag{30}$$

and

$$L(x, y, \lambda) = f(x, y) + \sum_{i \in I(x) \cup J} \lambda_i g_i(x, y) \tag{31}$$

is the Lagrangien of the lower level problem with respect to the actives constraints. The steepest descent then coincides with the optimal solution of the linear-quadratic bilevel program

$$\begin{aligned} &\min_d \nabla_x f(x, y^*) d + \nabla_y f(x, y^*) w(x, d) \\ &s.t\ ||d|| \leq 1, \\ &w(x, d)\ \textit{solves the quadratic problem}(), \end{aligned} \tag{32}$$

for which exact algorithms exists, such as those in [12].

Kolstad and Lasdon in [12] have adapted the previous computational method to large problem, where the lower-level problem may have hundred of x variables and/or constraints, and where many components of x have simple bounds. In addition, many of constraints in g may be inactive at the optimum. for extracting derivative information on the implicit function $y(x)$, which is especially efficient when the lower level problem has simple bounds on the variable and/or many inactive constraints. The quantity ∇x is computed using an adaptation of the methods and theory presented in Fiacco.

The third set of algorithms use double-penalty methods approximating both the lower level problem and upper level problem by sequences as unconstrained minimization problems of penalized augmented objective functions (see [16] and [17]).

1.1.6 Example of BLPP

1. **Agent Problem:** Some agency relations are Stackelberg games. Assume that there are N agents and a corporation and the agents are competitive or non-competitive. Let the cost function of corporation be $F(x, y)$ and the constraint be $G(x, y) \leq 0$. The cost function of the ith agent is $f_i(x, y)$ and the ith follower subjects constraints $g_i(x, y) \leq 0$. Then the model is

$$\begin{aligned} & min_x \;\; F(x,y) \\ & s.t \;\; G(x,y) \leq 0 \\ & \quad \text{where} \\ & y \;\; \in arg\ min_x \ (f_1(x,y), ..., f_N(x,y)) : \\ & \quad s.t \;\; g_i(x,y) \leq 0 \quad i = 1, ..., N \end{aligned} \tag{33}$$

1.2 Agent-Based Optimization

Agent Based Models (ABM) cope with optimization problems whose domains present several interrelated components characterized by some attributes, which interact in a distributed and heterogeneous environment.

In agent-based modelling (ABM), a system is modeled as a collection of autonomous decision-making entities called agent. In [18] Wooldridge and Jennings have defined an agent as a computational system interacting with an environment that characterized by the following features:

- Independence: The agent is free to act without any control of other entities.
- Social-ability: Each agent has to be able to communicate with the other entites in order the fulfil the goals.
- Re-activeness: Agents have the abilily to react over signals coming from the environment.
- Pro-activeness: Agents are endowed with goal-directed behaviors. They take the initiative in order to satisty their objectives.

The development of an ABM needs a complete description for a set of basic building blocks. According to [19], the previous maneuver needs an overview about the object of the simulation, the nature of agents' population as their degree of activeness and pro-activeness and finaly, the interaction paradigm among agents.

Each kind of interaction between two agents needs communication capabilities such as sending and receiving messages. Hence, it is necessary to establish a coordination mechanism among agents themselves, in order to avoid conflicts that can affect the achievement of their objectives. Two categories can be used:

- Distributed ABM. Agents has the control on themselves and are endowed with self-organizing rules for resource sharing and goal pursuing.

- Centralized ABM. Agents' behaviors are controled by a mediator agent who is assigned with the task of regulating.

As pointed out in [20], ABM can provide some advantages in terms of computational times, thanks to their ability to divide problems in several sub-problems, and tend to be preferable when the size of the problem is large, the domain is modular in nature and the structure of the domain changes frequently. However, these computational advantages can be offset by the need frequent interation in order to coordinate activities according to a given paradigm.

ABMs and classical heuristics have complementary characteristics. Hence, many studies focus in establishing approches that could link the both kind of optimization. In practice, optimization techniques are utilized for strategic planning and ABMs for operational and tactic re-planning.

Domain of application : Scheduling problem, transportation and logistics supplychain.

The most applications of ABMs deal with models under cooperative decentralize decision rules. Under these rules agents have to coordinate to perform best actions for a team. The problems involved by a such setting are known as ditributed constraint satisfaction problem (DisCSP) and distributed constraint optimization problem (DCOP).

1.2.1 Distributed Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) is a problem which deals with finding a consistent assignment of values to variables. A distributed CSP (DCSP) is a CSP in which variables and constraints are distributed among multiple automated agents. Its most common applications deal with configuration, planning, scheduling and ressource allocation and form the basis of a significant software industry.

Makoto Yokoo in [21] has made the following assymptions in order to solve the DCSP:

- Agents communicate by sending messages. An agent can send messages to other agents if the agent knows the addresses of the agents.

- The delay in delivering a message is finite, though random. For the transmission between any pair of agents, messages are received in the order in which they were sent.

Each agent has some variables and tries to determine theirvalues. However, there exist inter-agent constraints, and the value assignment must satisafy these inter-agent constraints. Formally, there exist m agents $1, 2, ..., m$. Each variable x_j belongs to one agent i (this relation is represented as $belongs(x_j, i)$). Constraints are also distributed among agents. The fact that an agent l knows a constraint predicate p_k is represented as $known(p_k, l)$.

A DCSP is solved if the following conditions are satisfied:

- $\forall\, i$, $\forall\, x_j$ where $belongs(x_j, i)$, the value of x_j is assigned to d_j, and $\forall\, l$, $\forall\, p_k$ where $known(p_k, l)$, p_k is true under the assignment $x_j = d_j$.

The following assumptions are made for simplificity in describing the algorithms :

- Each agent has exactly one variable, has a unique identifier.
- All constraints are binary.
- Each agent knows all constraint predicates relevant to its variable.

For an agent x_i, a set of agents reffers to each of which is directly connected to x_i by a link, as neighbors of x_i.

1.2.2 Asynchronous Backtracking Algorithm

Assyunchronous Backtracking is a asynchronous version of backtracking algorithm used as a standar method for solving Costraints Satistfaction Problems (CSPs) [21]. This algorthim is defined as following:

- Each agent is assigned a priority. Uasually, variables are subjected to an alphabetical order of their identifiers.
- Each agent instantiates its variable by selecting a random value from its domain.
- Each agent communicates its tentative variable assignments to its neighboring agents with taking into account the priority order.The message to communicate in this case is "ok?" (Agent with the higher priority, its assignment)". For example, the agent A has a higher priority than the agent B (alphabetical order). Lets A take the value 2. A send to B the message "$ok?(A = 2)$".

- The agent how received the message places the received assignment in a data called "agent-view".

- If the current value of the agent is consistent with his agent's view then he does nothing.

- If the current value of the agent is not consistent with his agent's view, the agent searches in his domain for a new consistent value:

  - If a consitent value exists, the agent assigns his variable that value and sen "ok?(agent,value)" to all his lower priority neighboors.

  - If no consistent value exists, the agent backtracks. This operation is executed by sending a message describing the conflict to the lowest priority agent whose variable is involved in the conflict. Then it waits for this agent to send back the new assignment to its variable.

    * If the agent receives a BACKTRACK message, but the conflict and the agent's view do not match, one of them must be obsolete; then it ignores the BACKTRACK.

    * Otherwise, it records the conflict as a new constraint must enforce. This operation requests a new link if necessary.

    * When an agent receives a NEW-LINK request, it adds the sender to its neighboring agents (children list) and responds through an OK? message.

    * If the agent receives a NO-SOLUTION message, it terminates.

  - If one of the conflicts set is empty set, this means any overset of is a conflict, so that there is no solution to the DSCP. The agent broadcasts a NO-SOLUTION message and terminates.

1.2.3 Asynchronous weak-commitment search

One limitation of the asynchronous backtracking algorithm is that the agent or variable ordering is statically determined. If the value selection of a higher priority agent is bad, the lower priority agents need to perform an exhaustive search to revise the bad decision. The asynchronous weak-commitment search algorithm [22] introduces the min-conflict heuristic to reduce the risk of making bad decisions. Furthermore, the agent ordering is dynamically changed so that a bad decision can be revised without performing an exhaustive search. The asynchronous weak-commitment search introduces the following priority system:

- For each agent, the initial priority is 0.

- If there no exists consistent value for x_i, the priority of x_i is changed to $k+1$, where k is the largest value of related agents.

- The order is defined such that any agent with a larger priority value has higher priority. If the priority values of agents are the same, the order is determined by the alphabatical order of the variables.

- As in the asynchronous backtracking, each agent concurrently assigns a value to its variable, and sends the variable value to other agents.

- The priority value, as well as the current assignment, is communicated through the "ok?" messages.

- If the current value is not consistent with the local view, the agent changes its value using the min-conflict heuristic, i.e., a value is consistent with the local view and minimizes the number of constraint violations with variable of lower priority agents.

2 Optimization in Uncertain environment

There is a various origins of uncertainty that could affect an aptimization problem. Beyer and Sendhoff class them in four principal classes:

- **Class A**: This class concerns the uncertainties due to environmental conditions, which can be represented as non-contollable parameters e of the model. Hence, the objective function can be represented by $f(d, e)$, where d is the vector of decision variables.

- **Class B**: It deals with inaccuracies on the decision variables. Indeed, it is impossible to build a real system with an infinite precision. Generally, the systems are built with a certain tolerance of the decision variables. Hence, the objective function of the system can be represented by $f(d + \delta, e)$, where δ is the disruption due to manufacturing.

- **Class C**: This class represents the measurement uncertainty of the system output. It is indeed impossible to accurately measure the output values and performance of the system. This class includes both uncertainty measurement errors and approximations errors made by the use of numerical model to represent the actual physical system. In practice, it is generally represented with a white noise.

- **Class D**: To have a feasible system, it is necessary to ensure that the constraints of the problem are satisfied. Similarly, it is difficult to acccurately evaluate the objective function f as it shown in teh previous classes. Hence, uncertainty can be on the constraints of the problem, which limit the set of parameters.

2.1 Stochastic Optimization

Stochastic programming deals with mathematical programs where some of the data incorporated into the objective or constraints is uncertain.

$$\begin{aligned} &\min_{x \in X} \quad g(x, \epsilon) \\ &s.t \quad g_i(x, \epsilon) \leq 0, \ i = 1, \cdots, m, \end{aligned} \tag{34}$$

where ϵ is a random vector.

In a such model, it is fondamental to know the distribution of the probability of the unkown parameters. Indeed, it is not always necessary to know the entire distribution. In general, having information about the impact of those variables is sufficient. Therefore, it must be able to identify the origine of the uncertainty. The most known appllication in this field copes with decisions making under uncertainty.

Example 1 (Newsvendor Problem) Many companies sell seasonal products, such as fashion articles, airline seats, Christmas decorations, magazines and newspapers. These products are characterized by a relatively short selling season, after which the value of the products decrease substantially. Often, a decision has to be made how much of such a product to manufacture or purchase before the selling season starts. Once the selling season has started, there is not enough time remaining in the season to change this decision and implement the change, so that at this stage the quantity of the product is given. During the season the decision maker may be able to make other types of decisions to pursue desirable results, such as to change the price of the product as the season progresses and sales of the product takes place. Such behavior is familiar in many industries. Another characteristic of such a situation is that the decisions have to be made before the eventual outcomes become known to the decision maker. For example, the decision maker has to decide how much of the product to manufacture or purchase before the demand for the product becomes known. Thus decisions have to be made without knowing which outcome will take place.

Stochactic programming approach requires the use of the "Scenario approach". This consists on approximation the randomness elements by some realization called scenarios. The most widely applied and studied sthocastic programming model is two-stage (linear) programs which can be extendes to a general form as Multistage model.

2.1.1 Two-stage model

This model is based on two stages of decision and are as following:

- A number of decisions have to be taken before the experiment. All these decisions are called first-stage decisions and the period when these decisions are taken is called the first stage.

- A number of decisions can be taken after the experiment. They are called second-stage decisions. The corresponding period is called the second stage.

The general form of two-stage stochastic programming problem is given by:

$$\min_{x \in X} \{g(x) = f(x) + E[Q(x, \zeta)]\} \tag{35}$$

whee $Q(x,\zeta)$ is the optimal value of the second-stage problem

$$\min_y\{q(y,\zeta)|T(\zeta)x+W(\zeta)y=h(\zeta)\} \tag{36}$$

The problem above can be reformulated as:

$$\min_{x\in A}\ g(x)=c^Tx+E[Q(x,\zeta)] \tag{37}$$

$$s.t \qquad Ax = b, \tag{38}$$

$$x \geq 0, \tag{39}$$

whee $Q(x,\zeta)$ is the optimal value of the second-stage problem

$$\min_q \qquad q(\zeta)^Ty, \tag{40}$$

$$s.t\ \ T(\zeta)x+W(\zeta)y=h(\zeta) \tag{41}$$

$$y \geq 0, \tag{42}$$

where $x\in A$ the first-stage decision vector, X is a polyhedral set, defined by a finite number of constraints, $y\in Y$ is the second stage decision vector and $\zeta(q,T,W,h)$ contains the data of the second stage problem. As the second-stage problem contains random data, its optimal value $Q(x,\zeta)$ is random variable. The distribution of this random variable depends on the first-stage decisions x, and therefore the first-stage problem cannot be solved without understanding the properties of the second-stage problem.

Assuming that W is deterministic and chosen such that system $Wy=\chi, y\geq 0$ have solution for every χ, the problem exhibits nice properties that have allowed for the design of some successful solution methods. For any realization $q(\zeta),\ T(\zeta),\ h(\zeta)$ the function $Q(x,\zeta)$ is piecewise linear and convex in x on the set: $\{x\in A,\ x\geq 0, Ax\geq b\}$. If the distribution of ζ is discrete, the function $Q(x,\zeta)$ is piecewise linear and convex on this set. In case is continuously distributed then $Q(x,\zeta)$ is convex and differentiable. Thus, we have a continuous convex programming problem under linear constraints.

The main difficulty in solving the two-stage problem remains in compution $E[Q(x,\zeta)]$. To overpass this problem, the approach of scenarios has been used. Assume that the random vector ζ has a finite number of possible realizations, called scenarios, say $\zeta_1,\cdots,\zeta_K$, with respective probability masses $p_1,\cdots,p_k$.Then the expectatiion in the first-stage problem's objective function can be written as the summation:

$$E[Q(x,\zeta)]=\sum_{k=1}^{K}p_kQ(x,\zeta_k) \tag{43}$$

For a given x, the expectation $\mathbb{E}[Q(x,\zeta_k)]$ is equal to the optimal value of the linear programming problem

$$\min_{x\in X} \qquad \sum_{k=1}^{K}\ p_k\ q_k^T\ y_k, \tag{44}$$

$$s.t \qquad T_kx+W_ky_k=h_k, \tag{45}$$

$$y_k\geq 0,\quad k=1,\cdots,K. \tag{46}$$

Thus, the two-stage problem can be formulated as one large linear programming problem:

$$\min_{x \in X} \quad c^T x + \sum_{k=1}^{K} p_k \; q_k^T \; y_k, \tag{47}$$

$$s.t \quad T_k x + W_k y_k = h_k, \; k = 1, \cdots, K \tag{48}$$

$$Ax = b, \tag{49}$$

$$x \geq 0, \quad y_k \geq 0. \tag{50}$$

The problem above is called the deterministic equivalent of the stochastic problem. Propreties of the expected recourse cost follow directly from properties of parametric linear programming. This linear program has a certain lock structure which makes it amenable to various decomposition methods such as L-shaped method developed by Van Slyke and Wets (Van Slyke-Wers, 1969).

This numerical approach works reasonably well if the number K of scenarios is not too large. Suppose, however, that the random vector ζ has m independently components each having just 3 possible realizations. Then the total number of different scenarios is $K = 3^m$. That is, the number of scenarios grows exponentially fast in the number m of random variables. In the case, even for a moderate number of random variables, say $m = 100$, the number of scenarios becomes so large that even modern components of ζ have continuous distributions.

The Sample Average Approximation (SAA) can be used in order to reduce the size of scenario set. It consist on generating a sample $\zeta_1, \cdots, \zeta_N$ of N replications of the random ζ. Each taken with the same probability $p_k = \frac{1}{K}$, the sample is asumedd to be independent identically distributed. Then the expectation $\mathbb{E}[Q(x, \zeta_k)]$ is approximated by the sample average $\hat{q}_N(x) = \frac{1}{N} \sum_{j=1}^{N} Q(x, \zeta_j)$. The first-stage problem is given by :

$$\hat{g}_N(x) = \min_{x \in X} \quad c^T x + \frac{1}{N} \sum_{j=1}^{N} Q(x, \zeta_j) \tag{51}$$

$$s.t \quad Ax = b \qquad x \geq 0. \tag{52}$$

More details and algorithms concerning numerical methods can be found in (Ermoliev-Wets, 1988).

The farmer's problem: Consider a European farmer who specializes in raising wheat, corn, and sugar beets on his 500 acres of land. During the winter, he wants to decide how much land to devote to each crop. The farmer knows that at least 200 tons (T) of wheat and 240 T of corn are needed for cattle feed. These amounts can be raised on the farm or bought from a wholesaler. Any production in excess of the feeding requirement would be sold. Over the last decade, mean selling prices have been \$170 and \$150 per ton of wheat and corn, respectively. The purchase prices are 40% more than this due to the wholesaler's margin and transportation costs. Another profitable crop is sugar beet, which he expects to sell at \$36/T; however, the European Commission imposes a quota on sugar beet production. Any amount in excess of the quota can be sold only \$10/T. The farmer's quota for next year is 6 000T. Based on past experience, the farmer knows that the mean yield on his land is roughly 2.5 T, 3 T, and 20 T per acre for wheat, corn, and sugar beets, respectively. Hence, they are the decisions on how many acres to devote to each crop.

Define the following variables:

- x_1 = acres of land devoted to wheat.
- x_2 = acres of land devoted to corn.
- x_3 = acres of land devotes to sugar beets.
- w_1 = tons of wheat sold.
- y_1 = tons of wheat purchased.
- w_2 = tons of corn sold.
- y_2 = tons of corn purchases.
- w_3 = tons of sugar beets sold at favirable price.

The problem is formulated as following:

$$Q(x,s) = min\{238y_1 - 170w_y + 210y_2 - 150w_2 - 36w_3 - 10w_4\} \tag{53}$$
$$s.t. t_1(s)x_1 + y_1 - w_1 \geq 200, \tag{54}$$
$$t_2(s)x_2 + y_2 - w_2 \geq 240, \tag{55}$$
$$w_3 + w_4 \leq t_3(s)x_3, \tag{56}$$
$$w_3 \leq 6000, \tag{57}$$
$$y, w \geq 0, \tag{58}$$

where $t_i(s)$ represents the yield of crop i under scenario s (or state of nature s).

2.1.2 Multistage model

Most practical decisions problems, however, involve a sequence of decisions that react to outcomes that envolve over time. The stochastic programming approach to a such problem is called Multistage programming. It's an extention of two-stage programming to a multi-stage setting. This model is based on a set of uncertain data $\zeta_1, \cdots, \zeta_T$ which is revealed gradually over time, in T periods and our decisions should be adapted to this process. The decisions process has the form

$$decision(x_1) \to observatin(\zeta_2) \to decision(x_2) \to \cdots observatin(\zeta_T) \to decision(x_T).$$

The sequence $\zeta_t \in \mathbb{R}^d$, $t = 1, \cdots, T$ of data vectors is a stochastic process, i.e, a sequence of random variables with specified probability distribution. Let $\zeta_{[t]} := (\zeta_1, \cdots, \zeta_t)$ be the history of the process up to time t. It may occure that the value of the decision vector x_t, chosen at stage t, depend on the information $\zeta_{[t]}$ available up to time t, but not on the results of the futur observations. Thus, x_t is a stochastic process.

In a generic form a T-stage stochastic programming problem can be written in the nested formulation

$$\min_{x_1 \in X_1} f_1(x_1) + \mathbb{E}[\min_{x_2 \in X_2(x_1\zeta_2)} f_2(x_2\zeta_2) + \mathbb{E}[\cdots + \mathbb{E}[\min_{x_T \in X_T(x_{T-1}, \zeta_T)} f_T(x_T, \zeta_T)]]] \tag{59}$$

It can be reformulated as following:

$$\min_{x_1, \cdots, x_T} \mathbb{E}[f_1(x_1) + f_2(x_2(\zeta_{[2]}), \zeta_2) + \cdots + f_T(x_T(\zeta_{[T]}), \zeta_T)] \tag{60}$$
$$s.t \;\; x_1 \in X_1, \; x_1(\zeta_{[t]}) \in X_t(x_{t-1}(\zeta_{[t-1]}), \zeta_t), \quad t = 2, \cdots, T. \tag{61}$$

Note that optimization in the previous problem is performed over implementable and feasible policies and that policies $x_2, \cdots, x_T$ are functions of the data process, and hence are elements of appropriate functional spaces, while $x_1 \in \mathbb{R}^n_1$ is a deterministic vector. Therefore, unless the data process $\zeta_1, \cdots, \zeta_T$ has a finite number of realizations, formulation (42) − (43) leads ti an infinite dimensional optimization problem.

Another possible way is to write the corresponding dynamic programming equations. That os, consider the last-stage problem

$$\min_{x_T \in X_T(X_{T-1}, \zeta_T)} f_T(x_T, \zeta_T) \tag{62}$$

The optimal value of this problem, denotes $Q_T(x_{T-1}, \zeta_T)$, depends on the decision vector x_{T-1}. At stage $t = 2, \cdots, T-1$, we formulate the problem

$$\min_{x_t} f_t x_t, \zeta_t + \mathbb{E}[Q_{t+1}(x_t, \zeta_{t+1})|\zeta_{[t]}] \tag{63}$$

$$s.t \quad x_t \in X_t(x_{t-1}, \zeta_t) \tag{64}$$

where $\mathbb{E}[\cdot|\zeta_{[t]}]$ denotes conditional expectation. Its optimal value depends on the decision x_{t-1} at the previous stage and realization of the data process $\zeta_{[t]}$, and denoted $Q_t(x_{t-1}, \zeta_t)$ (cost-to-go or value function) which is calculated recursively, going backyard in time. In the case where the process is Markovien, each cost-to-go funtion Q_t depends only on ζ_t. If moreover, the stagewise independence conditions holds, then each expectation function does not depends on realization of random process.

2.1.3 L-shaped method for multistage (and two-stage) stochastic linear programs

The basic idea of the L-shaped method is to approximate the nonlinear term in the objective of the problem (63)-(64) by successively appending supporting hyperplanes. The method is generalized to apply to problems with up to three periods and up to three hundred seventy-five different future scenarios.

In [23],the musltistage stochastic linear program considered by the algorithm is

$$\begin{aligned}
&\min \ c^1 x_1 + E_{\epsilon_2}[\min c^2 x_2 + \cdots + E_{\epsilon_T}[\min c_T x_T]] \\
&s.t \ \ A_1 x_1 = b_1, \\
&\quad B_1 x_1 + A_2 x_2 = \epsilon_2 \\
&\quad \ \vdots \\
&\quad B_{T-1} x_{T-1} + A_T x_T = \epsilon_T \\
&\quad x_t \geq 0, \ t = 1, ..., T, \epsilon_t \in E_t, \ t = 2, ..., T,
\end{aligned} \tag{65}$$

where c_t is a known vector in $\mathbb{R}^{n_t}$ for $t = 1, ..., T$, b_1 is a known vector in $\mathbb{R}^{m_1}$, ϵ_t is a random m_2-vector defined on the probability space $(\mathbb{E}_t, \mathbb{F}_t, F_t)$ for $t = 1, ..., T$, and A_t and B_{t-1} are correspondingly dimensioned known real-valued matrices."$\mathbb{E}_{\epsilon_t}$" denotes mathematical expectation with respect to ϵ_t.

The $L-$shaped method of Van Slyke and Wets [24] applies to (65) when $T = 2$. Methods for the multi-stage problem have generally assumed a specific structure for the problem. Beale, et al. [25] and Ashford [26] for example, consider a multistage production problem and implement an appropriate approximation. The generalization of the $L - shaped$ method introduced in Brige [27], does

not, however require any special structure except that the random vectors ϵ_t are discretely distributed.

The algorithm is called the Nested Decomposition for Stochastic Programming Algorithm (NDSPA). It is based on the observation that given a realization ϵ_t^j of the random vector in period t can be written (see Wets [28])

$$\begin{aligned} \min\ & c^t x_t^j + Q_{t+1}(x_t^j) \\ s.t\ & A_t x_t^j = \epsilon_t^j + B_{t-1} x_{t-1}^{a(j)} \\ & D_t^{l,j} x_t^j \geq d_t^{l,j},\ l = 1, ..., r_t^j, \\ & x_t \geq 0, \end{aligned} \tag{66}$$

where $q_{t+1}(x_t)$ is a convex function, $D_t^{l,j} \in \mathbb{R}^{n_t}$ for all l, $r_t^j \geq m_{t+1}$, and the constraint $D_t^{l,j} x_t^j \geq d_t^{l,j},\ l = 1, ..., r_t^j$ is a feasibility cut.

The previous program can be solved using a relaxed master problem (RMP):

$$\min \quad c_t x_t^j + \theta_t^j \tag{67}$$

$$s.t \quad A_t x_t^j = \epsilon_t^j + B_{t-1} x_{t-1}^{a(j)} \tag{68}$$

$$D_t^{l,j} x_t^j \geq d_t^{l,j},\ l = 1, ..., r_t^j \tag{69}$$

$$E_t^{l,j} x_t^j + \theta_t^j \geq e_z^{l,j},\ l = 1, ..., s_t^j \tag{70}$$

$$x_t^j \geq 0 \qquad . \tag{71}$$

Program (67) is solved to obtain $(\bar{x}_t^j, \theta_t^j)$. If $\bar{\theta}_t^j < Q_{t+1}(\bar{x}_t^j)$ then another optimality cut (70) is added to (67) then it is resolved . If $\bar{x}_t^j$ froces infeasibility in any future period then a feasibility cut (69) is added (67). This process is repeated until $\theta_t^j \geq Q_{t+1}(\bar{x}_t^j)$. For the construction of feasibility and optimality cuts, (Chapter 3 of [23]).

For implementation in multistage problems, it is assumed that there are a finite number K_t of scenarios in each period t. The scenarios consist of all possible realizationof the random vectors from periods 2 through t. For every period t scenarion j, there corresponds a unique ancestor scenario $a(j)$ in period $t-1$ and, perhaps, several descendant scenario $d(j)$ in period $t+1$. NDSPA solves (65) by first obtaining a feasible solution to (67)-(71) and for all t and j and by then sequentially solving (66) using the relaxaed master problem from periods T to one.

NDSPA

- **Step 0.**
  Solve (RMP) for $t = 1$ (dropping the scenario index j) where $\theta_1 = 0$, $r_1 = s_1 = 0$ and (68) is replaced by $A_1 x_1 = b_1$. Set $\theta_t^j = 0$ and $r_t^j = s_t^j = 0$ in (RMP) for all t and scenarios in j at t. (The indices r_t^j and s_t^j are updated whenever a constraint (69) or (70) is added to (RMP)).

- **Step 1.**
  If () is infeasible for $t = 1$, STOP. The problem (RMP) is infeasible. Otherwise, let $\bar{x}_1$ be the current optimal solution of (65) for $t = 1$. Use $\bar{x}_1$ as an input in (RMP) for $t = 2$. Solve (RMP) for $t = 2$ and all ϵ_2^j, $j = 1, ..., K_2$. If any period two problem (RMP) is infeasible, then add a feasibility cut (69) to (RMP) for $t = 1$, resolve (RMP) for $t = 1$, and return to 1. Otherwise

let $t = 2$ and go to 2.

- **Step 2.**

  1. Let the current period t optimal solutions be $\bar{x}_t^j$, $j = 1, ..., K_t$. Solve (RMP) for $t+1$ and all $j = 1, ..., K_{t+1}$ using the ancestor solution $\bar{x}_t^j$ in (68).
  2. If any period $t+1$ problem is infeasible, add a feasibility cut (69) to the corresponding ancestor period t problem and resolve that problem.
     If the period t problem is infeasible, let $t = t - 1$.
     If $t = 1$, go to Step 1.
     Otherwise, return to Step 2.1

  Otherwise, all period $t+1$ problems (RMP) are feasible.
  If $t \leq T - 2$, let $t = t + 1$ and return to Step2.1.
  Otherwise ($t = T - 1$), remove the $\theta_\tau^j = 0$ restriction for all periods τ and scenarios j at τ. Let the current value of each θ_τ^j be $\theta_\tau^j = -\infty$ if o constraints (70) are present. Go to Step 3.

- **Step 3.**

  1. Find $E_t^{l,j}$ and $e_t^{l,j}$ for a new constraint (70) at each scenario t problem (RMP) using the current period $t+1$ solutions.
  2. If there exists j such that
  $$\bar{\theta}_t^j < e_t^{l,j} - E_t^{l,j}\bar{x}_t^j, \tag{72}$$
  then add the new constraint (70) to each period t problem (RMP) for which (72) holds. Solve each period t problem (RMP). Use the resulting solutions $(\bar{x}_t^j, \theta_t^j)$ to form (68) for the corresponding descendant period $t+1$ problems (RMP) and resolve each period $t+1$ problem (RMP).
     - If $t < T - 1$, let $t = t + 1$ and go to Step 2.1.
     - Otherwise, return to Step 3,1.

  Otherwise, $\bar{\theta}_t^j = e_t^{l,j} - E_t^{l,j}\bar{x}_t^j$ for all scenarios j at t.
     - If $t > 1$, let $t = t-1$ and return to Step 3,1, Otherwise, STOP. The current solutions $\bar{x}_\tau^j$, $\tau = 1, ..., T$ form an optimal solution of (65).

2.2 Multiobjective Stochastic Linear Programming

In recent years, multiobjective stochastic programming problems have become increasingly important in scientifically based decision making involved in practical problem arising in economics, industry, health care, transportation, agriculture, military purposes, and technology. A multiobjective stochastic linear programming problem is defined as follow:

$$\min_{x \in D(w)} (c^1(w)x, ..., c^k(w)x) \tag{73}$$

where

$$D(w) = x \in \mathbb{R}^n : A(w)x \leq b(w); x \geq 0$$

where $(c^1(w)x, ..., c^k(w)x)$ are $n-$dimensionql rqndon vectors defined on a probability space $(\Omega, \Gamma, \mathbb{P})$, $A(w)$ and $b(w)$ are respectively $m \times n$ and $m \times 1$ random matrices defined on the same probability

space.

As an example of a concrete problem that may be put into the form of (73), there is the automated manufacturing system in a production planning situation, with several objective functions, where the costs and time of production are known stochastically [30].

2.2.1 Methodological Approaches for Solving Multiobjective Stochastic Linear Programs

For this method the following assumptions should be met: $A_i(w),\ i = 1, ..., m;\ b(w)$ and $c^k(w)$, $k = 1, ..., K$ are normally distributed random vectors. $\lambda_k,\ k = 1, ..., K$ are strictly positive real numbers in the interval $(0, 1]$ such that $\sum_{k=1}^{K} \lambda_k = 1$.Moreover, the following notations are used:

- $h_i(w, x) = A_i(w)x - b_i(w),\ i = 1, ..., m.$
- Φ denotes the cumulative distribution function if the standard normal random variable.
- q_1 and q_2 are weights associated with the expected value and the standard deviation of $c(w)$ respectively.
- $\alpha = (\alpha_1, ..., \alpha_m)$ where $\alpha_i,\ i = 1, ..., m$ are probability levels prescribed by the Decision maker for constraints satisfaction.

A stepwise description of the method is as follow:

- **Step 1.** Read $\lambda_k,\ k = 1, ..., K;\ c^k(w),\ k = 1, ..., K;\ h_i(w, x),\ i = 1, ..., m;\ \alpha_i,\ i = 1, ..., m.$
- **Step 2.** Find

$$c(w) = \sum_{k=1} K\lambda_k c^k(w) \tag{74}$$

- **Step 3.** Replace $D(w)$ by

$$D^* = \{x \in \mathbb{R}^n : E(h_i(w, x)) + \Phi^{-1}(\alpha_i)\sigma(h_i(w, x)) \leq 0, i = 1, ..., m; x \geq 0\} \tag{75}$$

- **Step 4.** Solve the mathematical program:

$$\min_{x \in D^*} (q_1 E(c(w)) + q_2 \sigma(c(w))) \tag{76}$$

  and let x^* be the solution.

- **Step 5.** Stop.

This algorithm transforms the original problem into a single objective problem, that has been put in the deterministic form (76), using the expected value model approach [31]. The solution x^* obtained in an expected value-standard deviation efficient solution for ptoblem (73) (see [33]).

2.2.2 Multiobjective Two-stage Stochastic Programming

A general model of multiobjective two-stage stochastic programming can be stated as follow:

$$max\ z^t = \sum_{j=1}^{n} c_j^t x_j - \mathbb{E}[\sum_{i=1}^{m} q_i^t |y_i|], \quad t = 1, \cdots, T, \tag{77}$$

$$s.t.\ y_i = b_i - \sum_{j=1}^{n} a_{ij} x_j,\ i = 1, \cdots, m_1, \tag{78}$$

$$\sum_{j=1}^{n} d_{ij} x_j \leq b_{m_1+i},\ i = 1, \cdots, m_2, \tag{79}$$

$$x_j \geq 0,\ j = 1, \cdots, n\ y_i \geq 0, i = 1, \cdots, m_1, \tag{80}$$

where $x_j, j = 1, \cdots n$ and $y_i,\ i = 1, \cdots, m_1$ are the first stage and second-stage decision variables respectively. Further, $q_i,\ i = 1, \cdots, m_1$ are defined as the penalty cost associated with the discrepancy between $\sum_{j=1}^{n} a_{ij} x_j$ and b_i and $\mathbb{E}$ is used to represent the expected value of the discrete random variable.

Remark The statistical approach requires that the distribution function of the data must be known in advance. Otherwise it will not lead to an optimal result. This can be a real problem in many cases since it is often difficult to obtain a statistical model of the data. For this reason the statis-tical approach is not suitable if no accurate model is known, which is usually the case in assessing customer demand for a product, for example. Estimation errors have especially dire consequences in industries with long production lead times, such as the automotive, retail and high-tech industries: They result in stockpiles of unneeded inventory or lost sales and customers' dissatisfaction. Moreover, another disadvantage of stochastic programming is that the size of the resulting deterministic model increases drastically as a function of the number of scenarios, making the running time exponential. These two draw-backs are the reason why not everyone just uses the statistical approach, but why some researchers have tried to come up with alternatives such as robust optimization. Research has further been motivated by a recent increase in an demand for such an alternative due to volatile customer tastes, technological innovation and reduced product life cycles.

2.3 Robust Optimization

Robust optimization concerns situations where different scenarios on data ae considered, and where the objective is to determine solution which remains "good" regardless the scenarios chosen. The idea is to build a flexible solution which can react to modifications on data with taking into account the uncertainty orthe impreicion on the costs. In the real world, models for environmental and energy systems assessments are plagued with uncertainty. They are usually formulated as mid-term to long-term planning problems, crucially influenced by uncertain factors like climate change, economy growth, technological progress.

2.3.1 Robust linear optimization

The robust conterpart of a linear optimization problem is written, without loss of generality, as

$$\begin{aligned} &\min && c^T x \\ &s.t && Ax \leq b,\ \forall a_1 \in U_1, \cdots, a_m \in U_m, \end{aligned}$$

where a_i represents the i^{th} row of the uncertain matrix A, and takes values in the uncertain set $U_i \subseteq \mathbb{R}^n$. Then, $a_i^T x \leq b_i$, $\forall a_i \in U_i$, if and only if

$$\max_{a_i \in U_i} \quad a_i^T x \leq b_i, \quad \forall i. \tag{81}$$

The set of the above subproblem determines the complexity of solving the Robust Oprimization problem.

Ellipsoidal Uncertainty Let U be "ellipsoidal," i.e $U = U(\Pi, Q) = \{\Pi(u) | \ ||Qu|| \leq \rho\}$, where $u \to \Pi(u)$ is an ffine embedding of $\mathbb{R}^{\mathbb{L}}$ into $\mathbb{R}/m \times n$ and $Q \in \mathbb{R}^{M \times n}$. According to Ben-Tal and A.Nemirovski's in [34], the previous robust counterpart problem is equivalent to a second-order cone program (SOCP). Explicitly, if

$$U = \{(a_1, \cdots, a_m) : a_i = a_i^0 + \Delta_i u_i, \ i = 1, \cdots, m, \ ||u||_2 \leq \rho\} \tag{82}$$

where a_i^0 denotes the nominal value. Then the robust counterpart is:

$$\min \quad c^T x \tag{83}$$
$$s.t \quad a_i^0 x \leq b_i - \rho ||\Delta_i x||_2 \quad \forall i = 1, \cdots, m. \tag{84}$$

The subproblem (81) is an optimization over a quadratic constraint. The dual, therefore, involves quadratic functions, which leads to the resulting SOCP.

Polyhedral Uncertainty When U is polyhedral, the subproblem becomes linear, and the the robust counterpart is equivalent to a linear optimization problem, To illustrate this, consider the problem:

$$\min \qquad c^T x \tag{85}$$
$$s.t \qquad \max_{\{D_i a_i \leq d_i\}} a_i^T x \leq b_i, \ \forall \ i = 1, \cdots, m, \tag{86}$$

With using the dual of the problem above the problem (85) − (86) becomes:

$$\min \quad c^T x \tag{87}$$
$$s.t. \ p_i^T d_i \leq b_i, i = 1, \cdots, m \tag{88}$$
$$p_i^T D_i = x, \quad i = 1, \cdots, m \tag{89}$$
$$p_i \geq 0, i = 1, \cdots, m. \tag{90}$$

The size of this problem grows polynomially in the size of the nominal problem and the dimensions of the uncertainty set.

In short, for many choices of the uncertainty set, robust linear optimization problems are tractable. Bertsimas and A. in [37] have studied the case where the unconstraited set is described by more general norms leas to convex problems with constraints related to the dual norm. Bertsimas and Sim in [36] consider an uncertainty sets that control the number of parameters of the problem that are allowed to vary from their nominal values, providing a different trade-off between the optimality of the solution, and its robustness to parameter perturbation.

2.3.2 Robust geometric programming

A geometric program (GP) is a convex optimization problem of the form

$$\min \quad c^T y \tag{91}$$
$$s.t\ g(A_i y + b_i) \leq 0, \quad i = 1, \cdots, m. \tag{92}$$
$$Gy + h = 0, \tag{93}$$

where $g : \mathbb{R}^k \to \mathbb{R}$ is the $log - sum - exp$ function, $g(x) = log(\sum_{i=1}^{k} e^{x_i})$, and the matrices and vectors A_i, G,b_i, and h are of appropriate dimension. For many engineering, design, and statistical applications of GP, see [39].

$$g(\tilde{A}_i(u)v + \tilde{b}_i(u)) \leq 0, \quad \forall u \in U \tag{94}$$

where $(\tilde{A}_i(u)v + \tilde{b}_i(u))$ are affinely dependent on the uncertainty u, and U is an ellipsoid or polyhedron. The complexity of this problem is unknown.

2.3.3 Robust discrete optimization

Bertsimas and Sim in [35] present a model for cost uncertainty in which each coefficient c_j is allowed to vary within the interval $[\bar{c}_j, \bar{c}_j + 1]$, with no more than $\Gamma \geq 0$ coefficients allowed to vary. They then apply this model to a number of combinatorial problems, i.e., they attempt to solve

$$\min \quad c^T x + \max_{S \subset D} \sum_{j \in S} d_j x_j \tag{95}$$
$$s.t \quad x \in X, \tag{96}$$
$$\tag{97}$$

where $N = 1, \cdots, n$, $D = \{S | S \subseteq \mathbb{N}, |S| \leq \Gamma\}$ and X is a fixed subset. They show that under this model for uncertainty, the robust version of a combinatorial problem may be solved by solving no more than $n + 1$ instances of the underlying, nominal problem. They also show that this result extends to approximation algorithms for combinatorial problems. For network flow problems, they show that the above model can be applied and the robust solution can be computed by solving a logarithmic number of nominal, network flow problems.

2.3.4 Optimization models with probabilistic constraints

Chance (or probabilistic) constrained optimization problem can be writhem in the following form :

$$\min_x \ E[f(x, \zeta)] \tag{98}$$
$$s.t \tag{99}$$
$$either\ P\{g_i(x, \zeta) \leq 0\} \geq \alpha_i, \ i = 1, \cdots, m. \tag{100}$$
$$or\ P\{g_i(x, \zeta) \leq 0\ i = 1, \cdots, m\} \geq \alpha. \tag{101}$$

where $f, g : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ are at least differentiable with respect to $x \in X \subset \mathbb{R}^n$, x is a vector of deterministic variables; $\zeta \subset \Omega \subset \mathbb{R}^p$ is a vector of random variables with joint probability density $\phi(\zeta)$. P is the probability measure induced by the random ζ.

Example In the following example probabilistic constraints arise in a natural way. Consider n investment opportunities, with random returns $R_1, \cdots, R_n$ in the next year. The initial capital K and the aim is to invest some of it in such a way that expected value of our investment after a year is maximized, under the condition that the chance of losing no more than a given fixed amount $b > 0$ is at least p, where $p \in (0, 1)$. Such a requirement is called the Value at Risk (VaR) constraint. Let $x_1, \cdots, x_n$ be the amounts invested in the n opportunities. The investment changes in value after a year by $g(x, R) = \sum_{i=1}^{n} R_i x_i$. The stochastic optimization problem with probabilistic constraints can be formulated as following:

$$\max \quad \sum_{i=1}^{n} \mathbb{E}[R_i] x_i \tag{102}$$

$$s.t. \quad P[\sum_{i=1}^{n} R_i x_i \geq -b] \geq p, \tag{103}$$

$$\sum_{i=1}^{n} x_i \leq K, \tag{104}$$

$$x \geq 0. \tag{105}$$

Let $Z := g(x, \zeta)$ be the random variable. x is feasible if $P\{g(x, \zeta) \leq 0)\} \geq \alpha$ holds with realibity α. Define $p(x) := P\{g(x, \zeta) \leq 0)$ Therfore, the feasible set of the previous problem is :

$$Q := \{x \in X |\ p(x) \ \geq \alpha\} \tag{106}$$

Many studies have discusses several properties of the chance constrained optimization problem, which deal with the contituity of a such problem those in [32].

The main difficulty in solving this problem remains in the computation of the chance constraint givien by

$$p(x) = P\{g(x, \zeta) \leq 0\} = \int_{\{\zeta \in \Omega \ | P\{g(x,\zeta) \leq 0\}} \phi(\zeta) d\zeta$$

which makes the CCOP a hard-problem.

In some special cases, where the the probability density function of Z is known, the chance constranit can be subtituted by a deterministic constraint of the form $q(x) \leq 0$, that the entire optimization problem can be handled as an ordinary nonlinear programming problem. Depending on the form of $g(x, \zeta)$, the explicit form of $q(x)$ may be difficult to obtain. Here some simple examples where the $q(x)$ can be writheen in its analatical form:

1. If $g(x, \zeta) = a^T x + b - \zeta$, where $\zeta \in \mathbb{R} \ \sim N(\mu, \sigma^2)$. Then :

$$P\{g(x, \zeta) \leq 0\} = P\{a^T x + b \leq \zeta\} \tag{107}$$

$$= 1 - P\{a^T x + b \geq \zeta\} \tag{108}$$

$$= 1 - P\{\frac{\zeta - \mu}{\sigma} \leq \frac{(a^T x + b) - \mu}{\sigma}\} \tag{109}$$

$$= 1 - \phi(a^T x + b) \tag{110}$$

The chance constraint is then equivalent to $1 - \phi(a^T x + b) \geq \alpha$ or $\phi^{-1}(1 - \alpha) - (a^T x + b) \geq 0$.

2. If $g(x,\zeta)$ is a linear transformation of ζ; $g(x,\zeta) = a(x)\zeta + b(x)$, with $\zeta \sim N(\mu,\sigma^2)$ (the case where stochastic variable ζ can be separated from the decision variable x) the deterministic equivalent can be found fairly easily by using the probability density function (pdf) of ζ to find a valu β, from $p = \int \cdots \int_{\beta}^{\infty} pdf(\zeta)d\zeta$ such that $\tilde{g}(x) \leq \beta \Rightarrow P\{g(x) \leq \zeta\} \geq p$. Consider for example, a constraint of the form

$$P\{\zeta(Mx+b) \leq a\} \geq p, \quad \zeta \sim N(\mu,\sigma^2) \tag{111}$$

$$P(\frac{\zeta^T(Mx+b) - \mu^T(Mx+b)}{\sqrt{(Mx+b)^T\sigma^2(Mx+b)}} \leq \frac{a - \mu^T(Mx+b)}{\sqrt{(Mx+b)^T\sigma^2(Mx+b)}}) \geq \alpha \tag{112}$$

Let $F^{-1}(p)$ is the value of the inverse cumulative distribution function of the standard normal distribution evaluated at p Hence the probabilistic constraint can be recast as the deterministic constraint:

$$\mu^T(Mx+b) \leq a - F^{-1}(p)\sqrt{(Mx+b)^T\sigma^2(Mx+b)}. \tag{113}$$

Generally, chance constraint is severely computational. Whenever this is the case, a natural course of actions is to look for tractable approximation of the chance constraint, i.e., for efficiently verifiable sufficient conditions for its validity. The first method consist on using Sample Average Approximation (AAA) (See [39]). Define the followinf function:

$$\mathbb{I}_{(0,\infty]}(g(x,\zeta)) = \begin{cases} 0, & if\ g(x,\zeta) > 0 \\ 1, & if\ g(x,\zeta) \leq 0. \end{cases} \tag{114}$$

The idea is to determine samples $\{\zeta_1, \cdots, \zeta_N\} \subset \Omega$ and replace the chance constrains with

$$p_N(x) = \frac{1}{N}\sum_{k=1}^{N} \mathbb{I}_{(0,\infty]}(g(x,\zeta_k)) \leq \alpha. \tag{115}$$

$p_N(x)$ is called Relative-frequency count for the satisfactio of $g(x,\zeta)$. This method maintains the convexity of the chance constrained optimization problen and avoids computation of multidimensional integrals. On the other hand, the CCOP becomes a non-smooth optimization problem, furthermore, to have a feasible solution the number of samples must be very large; the feasibility of the obtained solution of the CCOP is guaranted only when $N \to \infty$.

Another method consist on generating a sample (independent identically distributed) $\zeta_1, \cdots, \zeta_N$ of N replications on the distribution of ζ from Ω using Monte-Carlo method, then approximate the constraint $P(g_i(x,\zeta) \leq 0, i = 1, \cdots, m)$ by $g_j(x,\zeta_{(n)})$. In other words, robust optimization considers the worst-case problem, which is giving by:

$$\min_x \ \frac{1}{N}\sum_{k=1}^{N} f(x,\zeta_k) \tag{116}$$

$$s.t \ \ g(x,\zeta_k) \leq 0, k = 1, \cdots, N, \tag{117}$$

$$x \in X \tag{118}$$

In the case where $f(\cdot,\zeta)$ is convex with respect to $x \in \mathbb{R}^n$, then under mild additional conditionsm with the sample size N satisfying :

$$N \geq \frac{2n}{1-\alpha} ln(\frac{1}{1-\alpha}) + (\frac{2}{1-\alpha}) ln(\frac{1}{\alpha}) + 2n, \tag{119}$$

the optimal solution obtained from the previous problem is solution of the chance constrained problem(worst-case), with reliability α.

The advantage of this method remain in the facility of implementing and solving the resulting problem. On the other hand, for a high reliability level α, the number of scenarios required becomes very large. Indeed, the sample size as giving by the formula above grows linealy with n, which makes it difficult to apply the approach already to medium-size problems (with $\alpha = 0.09$ and $n = 200$ the estimate results is $N \geq 184\ 608$).

3 Optimization in Dynamic environment

Dynamic programming was the brainchild of an American Mathematician, Richard Bellman, who described the way of solving problems where it is needed to find the best decisions one after another. In the forty-odd years since this development, the number of uses and applications of dynamic programming has increased enormously. It is both a mathematical optimization method and a computer programming method. It is applicable to both discrete and continuous domains.It is a commonly used for solving complex problems by breaking them down into simpler problems. Furthermmore, it is a powerful technique for handling (characterizing and solving) problems with intertemporal decision-making. It allows to solve problems recursively, rather than at the beginning of time, while keeping track of all past histories in a relatively parsimonious way, using state variables. It is also applicable to problems that exhibit the properties of overlapping subproblems which are only slightly smaller and optimal substructure.

3.1 Dynamic Programming for discrete-time systems

For $N \in \mathbb{N}$, consider sequences $\{S_k\}_{k=0}^{N}$, $\{C_k\}_{k=0}^{N-1}$, and $\{D_k\}_{k=0}^{N-1}$ of (random) state spaces S_k, $0 \leq k \leq N$, control spaces C_k, $0 \leq k \leq N-1$, and (random) disturbance spaces D_k, $0 \leq k \leq N-1$. Given an initial state $x_0 \in S_0$, assume that the states $x_k \in S_k$, $0 \leq k \leq N$, evolve according to the discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad 0 \leq k \leq N-1,$$

where $f_k : S_k \times C_k \times D_k \to S_{k+1}$, $0 \leq k \leq N-1$. The controls u_k satisfy

$$u_k \in U_k(x_k) \subset C_k, \quad 0 \leq k \leq N-1,$$

i.e., they depend on the state $x_k \in S_k$, and the random disturbances w_k, $0 \leq k \leq N-1$, are characterized by a probability distribution

$$P_k(\cdot|x_k, u_k), \quad 0 \leq k \leq N-1$$

which may explicitly depend on the state $x_k \in S_k$, and the control u_k, but not on the prior disturbances w_j, $0 \leq j \leq k-1$.
At the time instants $0 \leq k \leq N-1$, decisions with respect to the choice of the controls have to made by means of contro laws

$$\mu_k : S_k \to S_k, \quad u_k = \mu_k(x_k), \quad 0 \leq k \leq N-1,$$

leading the control policy

$$\pi = \{\mu_0, ..., \mu_{N-1}\}.$$

where Π is the set of admissible control policies. The cost functionals are defined by

$$g_k : S_k \times C_k \times D_k \to \mathbb{R}, \quad 0 \leq k \leq N-1$$

associated with the decisions at time instants $0 \leq k \leq N-1$, and a terminal cost

$$g_N : S_N \to \mathbb{R}.$$

.

A discrete dynamic optimization problem is formulated as following:

$$\begin{aligned} \min_{\pi \in \Pi} \ J_\pi(x_0) &= E[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)], \\ s.t \ x_{k+1} &= f_k(x_k, \mu_k(x_k), w_k), \quad 0 \leq k \leq N-1 \end{aligned} \tag{120}$$

where the expectation E is taken over all random states and random disturbances. For a given initial state x_0, the value

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0), \tag{121}$$

then π^* is called and optimal policy.

3.1.1 Bellman's principle of optimality

Bellamn's principle of optimality is the key for a construcive approach to the solution of the minimization problem (120), which readlily leads to a powerful algorithmic tool: the Dynamic Programming algorithm (DP algorithm).

For the intermediate states $x_k \in S_k$, $0 \leq k \leq N-1$ that occur with positive probability, consider the minimization subproblems

$$\begin{aligned} \min_{\pi \in \Pi} \ J_\pi(x_0) &= E[g_N(x_N) + \sum_{l=k}^{N-1} g_k(x_l, \mu_l(x_l), w_l)], \\ s.t \ x_{l+1} &= f_l(x_l, \mu_l(x_l), w_l), \quad k \leq l \leq N-1 \end{aligned} \tag{122}$$

where $\pi_k = \{\mu_k, \mu_{k+1}, ..., \mu_{N-1}\}$ and Π_k is the set of admissible policies obtained from Π by deleting the admissible control laws associated with the previous time instants $0 \leq l \leq k-1$.

Bellman's optimality principle states that if

$$\pi^* = \{\mu_0^*, ..., \mu_{N-1}^*\}$$

is an optimal policy for the minization subproblem (120). Let

$$J_k^*(x_k) = J_{\pi_k^*}(x_k)$$

be the optimal cost-to-go for state x_k at the time instant k to the final time N. For completeness, set $J_N^*(x_N) = g_N(x_N)$. The intuitive justification for Bellman's optimality principle is that if the trancated policy π_k^* were not optimal for subproblem (122), then it's able to reduce the optimal cost for (120) by switching to the optimal solution for (122) once x_k is reached.

3.1.2 The DP algorithm and its optimality

Bellman's optimality principle strongly suggests to solve the optimality subproblems (122) backwars in time, beginning with the terminal cost at final time instant N and then recursively compute the optimal cost-to-go for subproblems $k = N-1, ..., 0$. This leads to the so-called backward DP algorithm which is characterized by the recursions

$$J_N(x_N) = g_N(x_N), \tag{123}$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k}[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))], \quad 0 \leq k \leq N-1, \tag{124}$$

where E_{w_k} means that the expectation is taken with respect to the probability distribution of w_k.

Theorem. (Optimality of the backward DP algorithm) Assume that

- The random disturbance spaces D_k, $0 \leq k \leq N-1$ are finite or countable sets.
- The expectations of all terms in the cost functionals in (124) are finite for every admissible policy.

Then, there holds

$$J_k^*(x_k) = J_k(x_k), \quad 0 \leq k \leq N. \tag{125}$$

Moreover, if there exist optimal policies μ_k^*, $0 \leq k \leq N-1$, for (124) such that $u_k^* = \mu_k^*(x_k)$, then $\pi^* = \{\mu_0^*, \mu_1^*, ..., \mu_{N-1}^*\}$ is an optimal policy for (120).

3.1.3 Applications of the DP algorithm

Consider the following inventory control problem:

The problem is to minimize the expected cost ordering quatities of a certain product in order in order to meet a stochastic demand for that product. The ordering is only possible at discrete time instants $t_0 < t_1 < \cdots < t_{N-1}$, $N \in \mathbb{N}$. Let x_k, u_k, and w_k, $0 \leq k \leq N-1$ be the available stock, the order and the demand at t_k. Here, w_k, $0 \leq k \leq N-1$, are assumed to be independent random variables. Here, the states x_k, the controls u_k and the demands w_k are non-negative integers which can take the values 0, 1 and 2 where the demand w_k has the same probability distribution

$$p(w_k = 0) = 0.1 \quad , \quad p(w_k = 1) = 0,7 \quad , \quad p(w_k = 2) = 0.2$$

for all planning periods $(k, k+1)$.Furthermore, assume that the excess demand $w_k - x_k - u_k$ is lost and that there is an upper bounf of 2 units on the stock that can be stored. Consequently, the equation for the evolution of the stock takes the form

$$x_{k+1} = \max(0, x_k + u_k - w_k)$$

under the constraint

$$x_k + u_k \leq 2.$$

For the holding costs and the terminal cost, assume that

$$r(x_k) = (x_k + u_k - w_k)^2 \quad , \quad R(x_n) = 0,$$

and suppose that the ordering cost is 1 per unit, i.e,

$$c = 1.$$

Therefore, the functions g_k, $0 \leq k \leq N$, are given by

$$\begin{aligned} g_k(x_k, u_k, w_k) &= u_k + (x_k + u_k - w_k)^2, \quad 0 \leq k \leq N-1, \\ g_N(x_N) &= 0. \end{aligned}$$

Finally, suppose $x_0 = 0$ and for the planning horizon, take $N = 3$ so that the recursions (122) of the backward DP algorithm heve the form

$$J_3(x_3) = 0, \tag{126}$$

$$J_k(x_k) = \min_{0 \leq u_k \leq 2 - x_k} E_{w_k}[u_k + (x_k + u_k - w_k)^2 + J_{k+1}(\max(0, x_k + u_k - w_k))], 0 \leq k \leq 2 \tag{127}$$

The execution of the algorithm will stard with period 2, followed by period 1, and finish with period 0:

- **Period 2:** Compute $J_2(x_2)$ for each possible value of the state x_2:

  1. $x_2 = 0$:

  $$\begin{aligned} J_2(0) &= \min_{u_2 \in \{0,1,2\}} E_{w_2}[u_2 + (u_2 - w_2)^2] \\ &= \min_{u_2 \in \{0,1,2\}} [u_2 + 0.1u_2^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2]. \end{aligned}$$

  The computation of the right-hand side for the three different values of the control u_2 yields

  $$\begin{aligned} u_2 = 0 \ : \ E[\cdot] &= 0.7 \cdot 1 + 0.2 \cdot 4 = 1.5, \\ u_2 = 1 \ : \ E[\cdot] &= 1 + 0.1 \cdot 1 + 0.2 \cdot 1 = 1.3, \\ u_2 = 2 \ : \ E[\cdot] &= 2 + 0.1 \cdot 4 + 0.7 \cdot 1 = 3.1, \end{aligned}$$

  Hence,

  $$J_2(0) = 1.3 \quad , \quad \mu_2^*(0) = 1.$$

  2. $x_2 = 1$: In view of the constraint $x_2 + u_2 \leq 2$ only $u_2 \in \{0, 1\}$ is admissible. Thus,

  $$\begin{aligned} J_2(1) &= \min_{u_2 \in \{0,1\}} E_{w_2}[u_2 + (1 + u_2 - w_2)^2] \\ &= \min_{u_2 \in \{0,1,2\}} [u_2 + 0.1(1 + u_2)^2 + 0.7u_2^2 + 0.2(u_2 - 1)^2]. \end{aligned}$$

  The computation of the right-hand side for the three different values of the control u_2 yields

  $$\begin{aligned} u_2 = 0 \ : \ E[\cdot] &= 0.1 \cdot 1 + 0.2 \cdot 1 = 0.3, \\ u_2 = 1 \ : \ E[\cdot] &= 1 + 0.1 \cdot 4 + 0.7 \cdot 1 = 2.1, \end{aligned}$$

  It follows that,

  $$J_2(1) = 0.3 \quad , \quad \mu_2^*(1) = 1.1.$$

3. $x_2 = 2$: Since the only admissible control is $u_2 = 0$, it follows

$$J_2(2) = E_{w_2}[(2 - w_2)^2] = 0.1 \cdot 4 + 0.7 \cdot 1 = 1.1,$$

hence

$$J_2(2) = 1.1 \quad , \quad \mu_2^*(2) = 0.$$

- **Period 1:** Compute again $J_1(x_1)$ for each possible value of the state x_1 taking into account the possible value $J_2(x_2)$:

1. $x_1 = 0$: then

$$J_1(0) = \min_{u_1 \in \{0,1,2\}} E_{w_1}[u_1 + (u_1 - w_1)^2 + J_2(\max(0, u_1 - w_1)]$$

The computation of the right-hand side for the three different values of the control u_2 yields

$$\begin{aligned} u_1 = 0 \ : \ E[\cdot] &= 0.1 \cdot J_2(0) + 0.7 \cdot (1 + J_2(0)) + 0.2 \cdot (4 + J_2(0)) = 2.8, \\ u_1 = 1 \ : \ E[\cdot] &= 1 + 0.1 \cdot (1 + J_2(1)) + 0.7 \cdot (1 + J_2(0)) + 0.2 \cdot (4 + J_2(0)) = 2.5, \\ u_1 = 2 \ : \ E[\cdot] &= 2 + 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot J_2(0) = 3.68. \end{aligned}$$

Hence,

$$J_1(0) = 2.5 \quad , \quad \mu_1^*(0) = 1.$$

2. $x_1 = 1$: then

$$J_1(1) = \min_{u_1 \in \{0,1\}} E_{w_1}[u_1 + (u_1 - w_1)^2 + J_2(\max(0, u_1 - w_1)]$$

The computation of the right-hand side for the three different values of the control u_2 yields

$$\begin{aligned} u_1 = 0 \ : \ E[\cdot] &= 0.1 \cdot (1 + J_2(1)) + 0.7 \cdot J_2(0) + 0.2 \cdot (1 + J_2(0)) = 1.5, \\ u_1 = 1 \ : \ E[\cdot] &= 1 + 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot J_2(0) = 2.68. \end{aligned}$$

Consequently,

$$J_1(1) = 1.5 \quad , \quad \mu_1^*(1) = 0.$$

3. $x_1 = 2$: Since the only admissible control is $u_1 = 0$, it follows that

$$\begin{aligned} J_1(2) &= E_{w_1}[(2 - w_1)^2 + J_2(\max(0, 2 - w_1)] \\ &= 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot J_2(0) = 1.68, \end{aligned}$$

hence

$$J_1(2) = 1.68 \quad , \quad \mu_1^*(2) = 0. \tag{128}$$

- **Period 0:** Here, only $J_0(0)$ is needed to be computed, since the initial state is $x_0 = 0$.

  1. $x_1 = 0$:

$$J_0(0) = \min_{u_0 \in \{0,1,2\}} E_{w_0}[u_0 + (u_0 - w_0)^2 + J_1(\max(0, u_0 - w_0)]$$

The computation of the right-hand side for the three different values of the control u_2 yields

$$\begin{aligned} u_1 = 0 \ : \ E[\cdot] &= 0.1 \cdot J_1(0) + 0.7 \cdot (1 + J_1(0)) + 0.2 \cdot (4 + J_1(0)) = 4.0, \\ u_1 = 1 \ : \ E[\cdot] &= 1 + 0.1 \cdot (1 + J_1(1)) + 0.7 \cdot J_1(0) + 0.2 \cdot (1 + J_2(0)) = 3.7. \\ u_1 = 2 \ : \ E[\cdot] &= 2 + 0.1 \cdot (4 + J_1(2)) + 0.7 \cdot (1 + J_1(1)) + 0.2 \cdot J_1(0)) = 4.818. \end{aligned}$$

Consequently,

$$J_0(0) = 3.7 \quad , \quad \mu_0^*(0) = 1.$$

Conclusion: The optimal ordering policy is to order one unit for each period, if the stock is 0, and to order nothing, otherwise.

Part III
Optimization Engine

1 Optimization Engine under Run Time

The next step is to build an Optimization Engine (OE) which has the capability to find an optimized solution with respect to some problems that could be fit in the previous discussed methods.

As discussed before, the conception of the SoS goes by the dynamic phase. In a such step, a platform is built in order to analyse, manage and control SoS operations. The figure bellow describes the platform components.

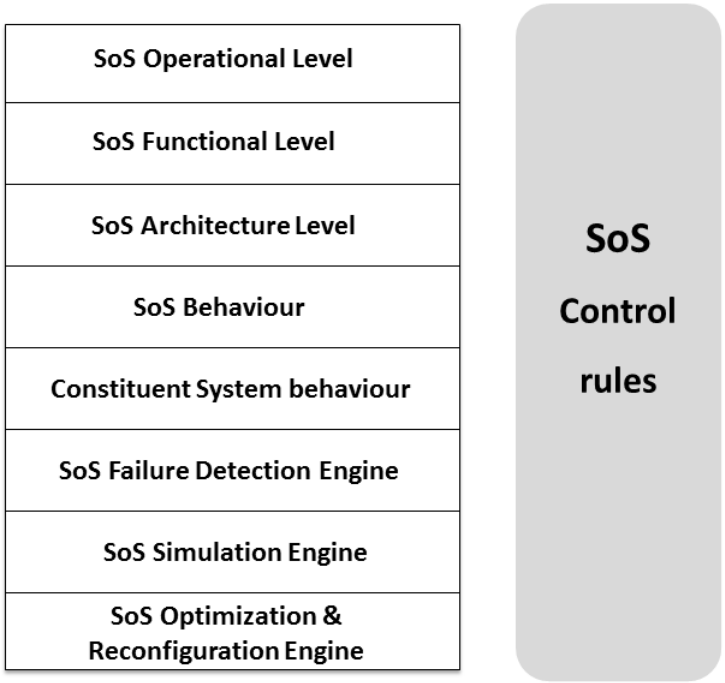

Figure 2: SoS Platform

The SoS operational, functional and architecture levels contain respectively discreptions about SoS operations, fuctionalities as welll as its architerture. SoS behaviour and constituent system behaviours gather information about the behavouir of the SoS and its constituents. The SoS failure detection engine helps in detecting and defining failure. This failure is reported to the SoS simulation and SoS optimization and reconfiguration engines in order to solve the problem and reconfurate the SoS. All this maneuver is done with respect to the SoS control rules.

As showed in the figure below, OE receives the inputs of the problem from the platform framework. OE has to define the most suitable algorithm to the problem and then resolve it. Afterwards, the solution is sent back to the framework in order to reconfigurate the systems.

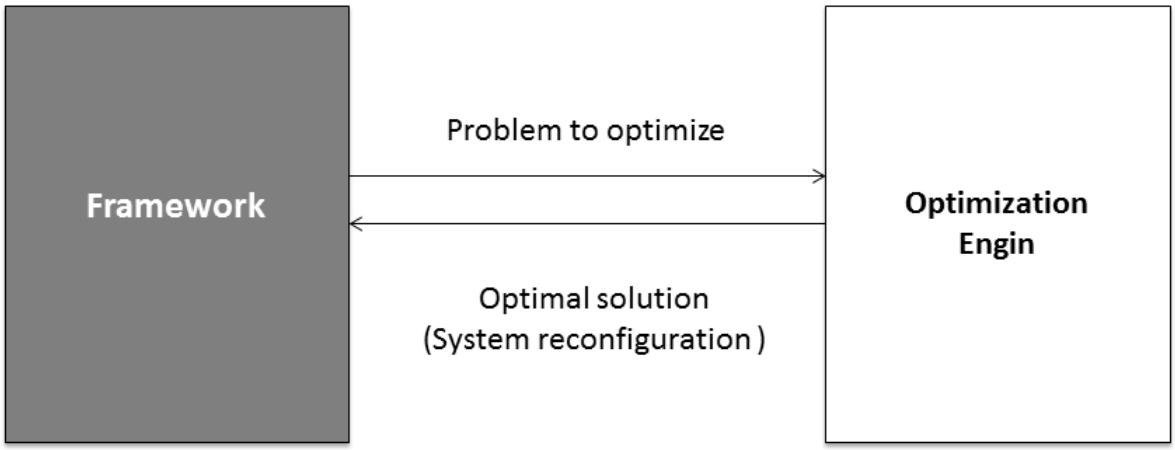

Figure 3: Optimization Engine

The OE is developed in MATLAB regarding to its flexibility. Indeed, SoS are developed using SysML (Systems Modeling Language). Systems are presented with blocks joined with links which define the nature of their interactions as well as relations. Once the SoS model is done, it is possible to simulate its behaviours. In some cases, simulations need complex tools such as advanced calculation tools. Hence, it is possible to add MATLAB to the SoS model (pattern).

OE algorithms are devided into three categories. The first one deals with optimization problems in a multi-player environment. The second one copes with optimization problems in uncertain environment and the third deals with the dynamic environment.

We have developed the algoritm for solving the Mixed-Integer Two-level programming problem, where we use MATLAB Optimization toolbox, which provides as with the linear programming solver "linprog". In the following section, we give an example of resolution of MITLPP.

2 Mixed Integer Two-Level Linear Programming Example

To verify the Branch and Bound algorithm for MITLPP, we take the following example from [13]:

$$
\begin{aligned}
(P1) \quad \max \quad & \{20x_1^1 + 60x_2^1 + 30x_3^1 + 50x_4^1 + 15x_2^1 + 10x_2^2 + 7x_3^2 : (x_1^1, x_2^1, x_3^1, x_4^1)\} \\
st: \quad & \max\{2x_1^2 + 6x_2^2 + 8x_3^2 : (x_1^2, x_2^2, x_3^2 | x_1^1, x_2^1, x_3^1, x_4^1)\} \\
& st: 5x_1^1 + 10x_2^1 + 30x_3^1 + 5x_4^1 + 8x_2^1 + 2x_2^2 + 3x_3^2 \leq 230 \\
& 20x_1^1 + 5x_2^1 + 10x_3^1 + 10x_4^1 + 4x_2^1 + 3x_2^2 \leq 240 \\
& 5x_1^1 + 5x_2^1 + 10x_3^1 + 5x_4^1 + 2x_2^1 + x_3^2 \leq 90 \\
& x_j^1 \in \{0, 1\}, \quad j = 1, 2, 3, 4. \\
& x_j^2 \geq 0, \quad j = 1, 2, 3.
\end{aligned}
$$

Let

$$
A^1 = \begin{pmatrix} 5 & 10 & 30 & 5 \\ 20 & 5 & 10 & 10 \\ 5 & 5 & 10 & 5 \end{pmatrix}, \; A^2 = \begin{pmatrix} 8 & 2 & 3 \\ 4 & 3 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 230 \\ 240 \\ 90 \end{pmatrix},
$$

$$
c^{11} = \begin{pmatrix} 20 & 60 & 30 & 50 \end{pmatrix}, \; c^{12} = \begin{pmatrix} 15 & 10 & 7 \end{pmatrix}
$$

$$
c^{22} = \begin{pmatrix} 2 & 6 & 8 \end{pmatrix}
$$

Once the algorithm is chosen in the OE. To run it, we need ta enter the inputs A^1, A^2, b, c^{11}, c^{12}, and c^{22}.

After solving the problem (F) (see the Mixed-Integer Two-level programming algorithm), we have $x_2^* = (0, 80, 23.333)$, $Z^* = 666.6667$. The resolution of the problem (B) leads to $Y_B^* = (2.3333; 1.778; 0)$, $Z_B^* = 963.333$ and $H = (-27.2222; 27.7778; -57.778; 20.5556)$. The following figure resumes all the resolution steps made by the algorithm.

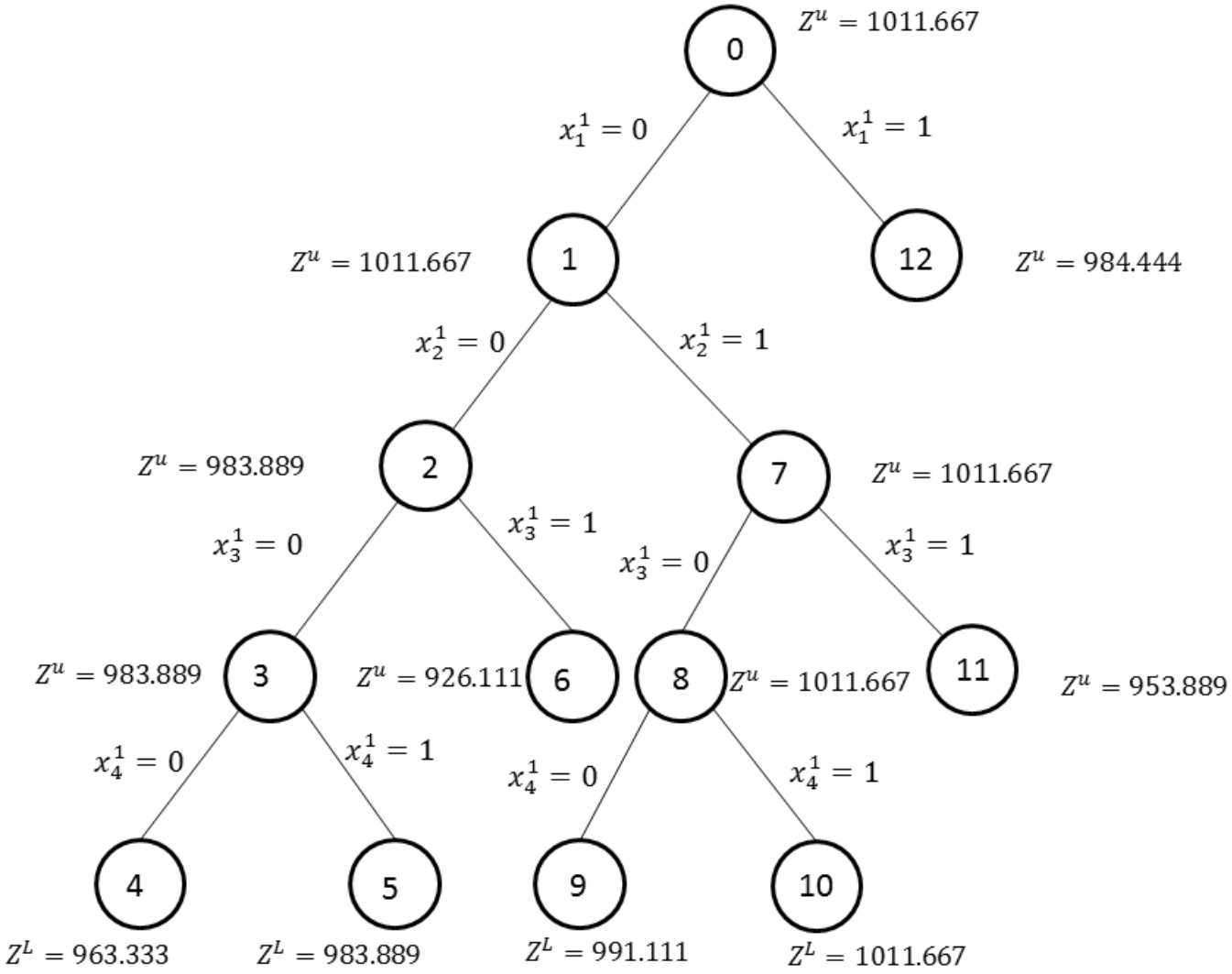

Figure 4: Branch-and-Bound tree for the example (P1)

In each node (iteration k), we check if $Z^u \leq Z^*$, and then we decide which node to branch upon next. The solution of the problem is $(x^{1*}, x^{2*}) = (0, 1, 0, 1; 0, 75, 21.667)$ with the high level optimal objective $Z^* = 1011.667$. The number of node generated with this algorithm is 12 which is less than the total node number of complete tree (31). This is due to the branching condition ($Z^u \leq Z^*$). The nodes 6, 11 and 12 are fathomed since the branching condition is not respected. The nodes 4, 5, 9 and 10 are terminated.

When the number of high-level variables increases, the execution time of exact algorithm increases approximately exponentially; the fewer low-level variables there are, the slower execution time grows. In [13], Wen and Yang have proposed a heuristic solution procedure, which can provide satisfactory near-optimal solutions in a reasonably short computational time.

2.1 Application to Command Control Center

In SoS, it is possible to describe communication systems problem as a Mixed-Integer Programming Problem. Assume that we have n_1 communication systems. Each system has its own cost and coverage rate. Furthermore, assume that we have n_2 constituent systems that we want to connect to each other through communication systems. Suppose that each constituent system has its own connection time as well as signal quality and realibility rate. Note that when the coverage rate increases, the signal quality as well as systems realibilities increase too. On the other hand, when the time increases, the realibility decreases. Taking into account these informations, the idea of this problem is to maximize - as the high level objective- the coverage rate of communication systems in addition to the quantity signal quatity×connection time, taking into account the maximazation of the quatity rability×connection time.

Let c^{11} be a vector with n_1 elements which refer to the coverage rate of each communication system. c^{12} is a vector of n_2 elemnts which refer to the signal quality value of each constituent system. c^{22} is a vector of n_2 elements which refer to the constituent system reability. Let A^2 be the cost matrix of communication systems and A^2 the cost matrix of connection time. Let b be the upper bound of the cost. Hence the problem can be formulated as in (MITLP).

Part IV
Conclusion and Further Outlooks

This report resumes the most common optimization methods that could fit in the system-of-systems. The SoS characteristics have many impacts on its operations. Hence, many challenges appear such as controlling SoS and constituent systems purposes and predicting emergent behaviours. In the literature, we found many optimization methods applied to multi-players environment. These methods deal with problems which exhibit hierarchical traits according to decision makers. We have stated the most common method in a this field such as Bi-level optimization method. On the other hand,to deal with the emergent behavior, the wide studied optimization problem in this field is the Two-stage model which can be solved with the L-shaped method.

The optimization engine developed here has for objective to find the best solution according to some criteria given as inputs. Until now, we have developed only two algorithms. The next step is to add the other algorithms which were introduced in the previous sections. The efficiency of an algorithm takes into account the nature of the solution as well as many other issues such as the computation time. Hence, we may add some modifications to these algorithms in order reduce the run time. This procedure is essential to make optimization under run time.

During my internship in EADS Innovation Works, I had the opportunity to enhance my knowledge in Optimization techniques as well as in System engineering field. To be a part of an important project allowed me to confront my ideas with the realities in the company. It brought to me a better visibility of the project organization in a long-term. Indeed, the exchanges with my colleagues allowed me to understand better the system engineering environment, goals and challenges. The participation in the meetings also helped me to seize the stakes in the project.

The work within Innovation Works needs to keep up to date with other research being carried out in, or related to, the field studied. This includes reading lots of articles as well as attending scientific meetings and conferences. Furthermore, the international profile of EADS helps me in developing my communication skills as well as integration issues within a multi-national team.

My internship in EADS Innovation Works was extremely enriching on the professional level as the personal. So, I got acquainted with the Optimization field, which was abstracted in the beginning. Within this company, I worked beside professionals experimented engineers and also beside students, who made their first step in the company as me. We all had in common the team spirit, sharing knowledge as well as the will to progress in a very rich innovation universe while being a part of huge projects. The liberties of action and management which were granted to me by my tutor make me more exigent towards my work.

Basing on this experience, I would like to continue working in the same field because I appreciate the research subject of my internship. Furthermore, I would like to work within a big company like EADS because every step of the project process is structured and well informed, so that allows employees to understand the evolution of the project and the importance of their contributions.

References

[1] Michael Masin and Evgeny Shindin, " SoS Optimization ", 2012.

[2] http://www.eads.com

[3] INCOSE, Systems Engineering Handbook, version 3.2.1 Juanuary 2011.

[4] M. Maier, "Architecting principles for systems-of-systems," Systems Engineering, vol. 1, no. 4, pp. 267-284, 1998.

[5] I. Sanduka, T. Lochow and R. Obermaisser, "Runtime Fault Prediction and Prevention for Emerging Services in System of Systems ".

[6] Stackelberg, H. The theory of market economy. Oxford: Oxford University Press, 195.2

[7] R. Cassidy, M. J. Kirby, and W. M. Raike, "Efficient distribution of ressources through three levels of government," Management Sci., vol. 17, no. 8, pp. 462-473, 1971.

[8] T. Bassar, Dynamic Games and Applications in Economics. New-York: Springer-Verlag, 1986.

[9] http://www.math.ucla.edu/ tom/LP.pdf

[10] Wayne F. Bialas and Mark H. Karwan, Two-Level Linear Programming, Management Science, Vol. 30, No. 8, August 1984.

[11] Bialas, W. F. and M. N. Chew, "A Game-Theoretic Approach to coalition Formation in Multilevel Decision-Making Organizations," Technical Report, Operations Research Program, Dept. of Industrial Eng., State University of NewYork at Buffalo, 1982.

[12] Jonathan F. Brad and James T. Moore, A Branch and Bound algorithm for the Bilevel Programming Problem, SIAM J. Sci. Stat. Comput, Vol. 11, No. 2, pp. 281-292, March 1990.

[13] U. P. Wen and Y. H. Yang, "Algorithms for solving the Mixed Integer Two-Level Linear Programming Problem," Computers Opns Res. Vol. 17, No. 2, pp. 133-142, 1990.

[14] Thomas A. Edmunds and Jonathan F. Bard, "Algorithms for Nonlinear Bilevel Mathematical Programs," IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, No. 1, January-February 1991.

[15] Kolstad and Lasdon, Derivative estimation and computational experience with large bilevel mathematical programs. Journal of Optimization Theory and Applications 65, 1990, 485-499.

[16] Aiyoshi, E., Shimizu, K. Hierarchical decentralized systems and its new solution by a barrier method. IEEE Transactions on Systems, Man, and Cybernetics, 11, 444– 449, 1981.

[17] Aiyoshi, E., Shimizu, K. A solution method for the static constrained Stackelberg problem via penalty method. IEEE Transactions on Automatic Control, 29, 1111– 1114, 1984.

[18] Wooldridge, M. and Jennings, N.R., 1995, "Intelligent agents: theory and practice," The knowledge Engineering Review, 10(2), pp. 115-152.

[19] Billari F. G., Fent, T., Prskawetz, A., and Sceffran, J. (Eds.). (2006). Agent-based computational modelling: Applications in demography, social, economic and environmental sciences (contributions to economics). Heidelberg: Physica-Verlag.

[20] Davidsson, P., Henesey, L., Ramstedt, L., Tornquist, J., and Wernstedt, F. (2005). Agent-based approaches to transport logistics. Transportation Research Part C: Emerging Technologies, 13 (4), 255-271.

[21] Makoto Yokoo and Katsutoshi Hirayama, Algorithms for Distributed Constraint Satisfaction: A review, Autonomous Agents and Multi-Agent Systems, 3, 185-207, 2000.

[22] M. Yokoo, "Asynchronous weak-commitment search for solving constraint distributed satisfaction problems," in Porc. First Int. Conf. Principles and Practice of constraint Programming (CP-95), Lecture Notes in Computer Science 976, 1995, pp. 88-102.

[23] Yu. Ermoliev and R. J-B Wets, "Numerical Techniques for Stochastic Optimization," Springer Series in Computational Mathematics, Chapter 12.

[24] R. Van Slyke and R.J-B. Wets, "L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Linear Programs," SIAM Journal on Appl. Math. 17(1969), 638-663.

[25] E.M.L. Beal, J.J.H. Forrest, and C.J. Taylor, "Multi-time Period Stochastic Programming," in Stochastic Programming, M.A.H. Dempster (ed.). Academic Press, New York, 1980.

[26] R.W. Ashford, "A Stochastic Programming Algorithm for Production Planning," Science report, Milton Keynes, 1982.

[27] J.R. Birge, "Solution Methods for Stochastic Dynamic Linear Programs," Technical Report SOL 80-29, Systems Optimization Laboratory, Standford University, 1980.

[28] J.R. Birge, " Decomposition and Partitioning Methods for Multi-Stage Stochastic Linear Programs," Technical Report 82-6, Department of Industrial and Operations Engineering. The University of Michigan 1982.

[29] R. J-B Wets, "Programming Under Uncertainty: The solution Set," SIAM Journal on Appl. Math. 14(1966), 1143-1151.

[30] H. Fazlollahtabar and I. Mahdavi, "Applying Stochastic Programming for Optimizing Production Time and Cost in an Automated Manufacturing System," International Conference on Computers and Industrial Engineering, Troyes, 6-9 July 2009, pp. 1226-1230.

[31] P.Kall and S.W. Wallace, "Stochastic Programming," John Willey and Sons Inc, New York, 1997.

[32] Birge, J. R. and Louveaux, F. Introduction to Stochastic Programming. Springer Series in Operations Research, Springer-Verlag, New York, NY, 1997.

[33] A. Segun Adeyefa and Monga K. Luhandjula, "Multiobjective Stochastic Linear Programming: An overview", American Journal of Operations Research, 2011,1,203-213.

[34] Ben-Tal, A., Nemirovski, A. Robust Convex Optimization. Math. of Oper. Res.,23:4, 769-805, 1998.

[35] D. Bertsimas and M. Sim., "Robust discrete optimization and network flows," Mathematical Programming Series B, 98:49-71, 2003.

[36] D. Bertsimas and M. Sim., "The price of robustness," Operations Research, 52(1):35-53, 2004.

[37] D. Bertsimas and A. Thiele., "A robust optimization approach to supply chain management," Operations Research, 54(1):150-168, 2006.

[38] K.-L. Hsiung, S.-J. Kim, and S. Boyd., "Tractable approximate robust geometric programming," Submitted to Mathematical Programming, 2005.

[39] S. Boyd and L. Vandenberghe, " Convex Optimization," Cambridge University Press, 2004.

[40] Verweij, B., Ahmed, S., Kleywegt, A., Nemhauser, G., and Shapiro, A., "The Sample Average Approximation method applied to stochastic routing problems: A computational study," Computational Optimization and Applications, 24(2-3), 289-333, 2003.