Priorització de comandes a SAP Business One

Martí Vancells Borralleras
Dimarts, 28 de juny de 2022

Treball Final de Grau vinculat amb l'empresa Seidor

Director: Joan Jiménez Martín
Ponent: Xavier Franch Gutiérrez
Departament del ponent: ESSI - UPC
Grau en Enginyeria Informàtica
Especialitat de Computació
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Resum

La gestió dels estocs és un tema important per les empreses. Disposar de molt estoc per un nombre gran de productes que asseguri poder satisfer les demandes diàries pot tenir un alt cost d'emmagatzematge i altres costos associats especialment en productes que caduquen i en moments en que hi ha grans fluctuacions en la demanda. Les empreses intenten aconseguir un equilibri en la gestió dels estocs per aconseguir la màxima competitivitat intentant servir als seus clients en un temps adequat i evitant alhora grans costos d'emmagatzematge. Quan l'estoc és insuficient per cobrir les demandes diàries, cal seleccionar quines de les comandes rebudes es serveixen i quines queden a l'espera de l'arribada de més estoc. L'empresa que ha sol·licitat aquest projecte vol prioritzar les comandes a servir tenint en compte les característiques dels clients i de la pròpia comanda.

Així doncs, el problema que se'ns planteja és el següent: A partir d'un conjunt de comandes i unes restriccions d'estoc dels productes que hi apareixen, es vol prioritzar l'assignació de l'estoc disponible sobre el conjunt de comandes. Aquesta priorització ve marcada per uns criteris parametritzables els quals fan referència al tipus de client que fa la comanda, així com a algun atribut de la pròpia comanda.

En aquest projecte, es fa un estudi sobre quina tècnica és la més adequada per a resoldre el problema pel tipus de comandes i perfil de clients de l'empresa, considerant des d'algoritmes que permeten trobar una solució aproximada fins a algoritmes que troben una solució òptima.

La tècnica escollida és la que s'implementa al sistema SAP de l'empresa. A partir dels criteris de priorització escollits, es realitza una puntuació de les comandes i s'obté la selecció de comandes a servir que aconsegueix la millor puntuació satisfent les restriccions d'estocs. La solució genera una distribució dels estocs dels magatzems per cadascun dels productes de les comandes seleccionades.

Resumen

La gestión de stocks es un tema importante para las empresas. Disponer de mucho stock para un gran número de productos que asegure poder satisfacer las demandas diarias puede tener un alto coste de almacenamiento y otros costes asociados, especialmente en productos que caducan y en momentos en que existen grandes fluctuaciones en la demanda. Las empresas persiguen un equilibrio en la gestión de los stocks para conseguir la máxima competitividad intentando servir a sus clientes en un tiempo adecuado y evitando al mismo tiempo grandes costes de almacenamiento. Cuando el stock es insuficiente para cubrir las demandas diarias, es necesario seleccionar cuáles de los pedidos recibidos se sirven y cuáles quedan a la espera de la llegada de más stock. La empresa que ha solicitado este proyecto desea priorizar los pedidos a servir teniendo en cuenta las características de los clientes y del propio pedido.

Así pues, el problema que se nos plantea es el siguiente: a partir de un conjunto de pedidos y unas restricciones de stock de los productos que aparecen, se quiere priorizar la asignación del stock disponible sobre el conjunto de pedidos. Esta priorización viene marcada por unos criterios parametrizables que hacen referencia al tipo de cliente que realiza el pedido, así como a algún atributo del propio pedido.

En este proyecto, se realiza un estudio sobre qué técnica es la más adecuada para resolver el problema por el tipo de pedidos y perfil de los clientes de la empresa, considerando desde algoritmos que permiten encontrar una solución aproximada hasta algoritmos que encuentran una solución óptima.

La técnica escogida es la que se implementa en el sistema SAP de la empresa. A partir de los criterios de priorización escogidos, se realiza una puntuación de los pedidos y se obtiene la selección de pedidos a servir que consigue la mejor puntuación satisfaciendo las restricciones de stock. La solución genera una distribución de los stocks de los almacenes por cada producto de los pedidos seleccionados.

Abstract

Stock management is an important issue for companies. Holding a lot of stock for a large number of products to ensure that daily demands can be met can have high storage and other associated costs, especially for products that expire and at times when there are large fluctuations in demand. Companies try to strike a balance in stock management to achieve maximum competitiveness by trying to serve their customers in a timely manner while avoiding high storage costs. When the stock is insufficient to cover the daily demands, it is necessary to select which of the received orders are served and which are left to await the arrival of more stock. The company that has requested this project wishes to prioritize the orders to be served, taking into account the characteristics of the customers and of the order itself.

Therefore, the problem that arises is the following:Starting from a set of orders and some stock restrictions of the products, we want to prioritize the allocation of the available stock over the set of orders. This prioritization is marked by parameterisable criteria that refer to the type of customer placing the order, as well as to some attribute of the order itself.

In this project, a study is carried out on which technique is the most appropriate to solve the problem by the type of orders and the profile of the company's customers, considering algorithms that find an approximate solution and algorithms that find an optimal solution.

The chosen technique is the one that is implemented in the company's SAP system. Based on the chosen prioritization criteria, the orders are scored, and the selection of orders to be served that achieve the best score while satisfying the stock constraints is obtained. The solution generates a distribution of the warehouse stocks for each product in the selected orders.

Índex

1. Context

1.1. Introducció

El treball de fi de grau *"Prioritització de comandes a SAP Business One"* pertany a l'especialitat de computació del Grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona (FIB). Aquest ha estat realitzat a *Seidor*, empresa on l'autor d'aquest projecte ha cursat les pràctiques curriculars.

1.2. Seidor i l'ERP SAP Business One

Seidor [1], l'empresa on s'ha realitzat aquest treball de final de grau, és una consultora tecnològica que ofereix un portafoli integral de solucions i serveis d'innovació, customer experience, ERP, analytics, employee experience, cloud, workplace i ciberseguretat, ideades perquè les organitzacions superin els seus reptes i aconsegueixin els seus objectius.

Seidor compta amb una unitat de negoci especialitzada en serveis end-to-end en l'ecosistema SAP: consultoria, venda de programari i implantació i manteniment d'aplicacions i infraestructures.

Una de les solucions SAP que implementa Seidor és l'ERP SAP Business One [2] per a empreses petites i mitjanes, una solució ERP completa i integrada que cobreix les necessitats de totes les àrees del negoci.

Els processos de negoci inclosos a SAP ERP són:

- Operacions
    - Distribució i Vendes
    - Gestió de Materials
    - Planificació de Producció
    - Logística
    - Gestió de Qualitat
- Finances
    - Comptabilitat Financera
    - Comptabilitat de Gestió

- Administració de Recursos Humans
    - Formació
    - Nòmina
    - Contractació
- Serveis Corporatius
    - Administració de Viatges
    - Medi Ambient
    - Salut
    - Seguretat

L'ERP SAP Business One aconsegueix automatitzar, integrar i agilitzar processos eficientment. Tanmateix, també compta amb un flux d'informació segura i un fàcil accés a dades rellevants, les quals permeten una factible generació d'anàlisis i informes per a efectuar una presa de decisions ràpida i intel·ligent.

Així doncs, Seidor completa el programa de SAP, afegint les funcionalitats que els clients necessiten i/o demanen, per a una millora de les seves gestions empresarials.

1.3. Problema a resoldre i conceptes tècnics

Des de Seidor hi ha un client que demana un desenvolupament on, donat un gran nombre de comandes de diferents clients que els arriben a SAP en format JSON provinents d'un servei web, vol prioritzar els clients als quals assigna l'estoc en cas de no tenir estoc suficient d'algun dels articles que apareix a les comandes.

Al tractar-se d'una selecció sobre un conjunt que ha de satisfer una sèrie de prioritats i restriccions podem reduir el problema principal d'aquest projecte a un problema d'optimització. Aquest problema es pot enfocar de diverses maneres, des d'implementar algoritmes d'optimització per a trobar una solució òptima fins a buscar una solució aproximada. S'ha realitzat un estudi per decidir quina és la millor opció pel tipus de comandes que rep l'empresa client.

1.4. Actors implicats

En aquest projecte hi ha diversos actors implicats:

- *Desenvolupador del projecte*: L'autor d'aquest projecte ha estat l'encarregat de desenvolupar-lo i de procurar aconseguir un resultat de qualitat.
- *Supervisor del projecte*: El director d'aquest projecte ha estat l'encarregat de validar la feina que el desenvolupador ha anat fent.
- *Tutor del projecte*: El ponent d'aquest projecte ha estat l'encarregat de supervisar que el TFG es faci d'acord als estàndards acadèmics de la FIB.
- *Seidor:* L'empresa on s'ha desenvolupat el treball de fi de grau vol satisfer les necessitats del client.
- *Empresa client:* El producte està dirigit a una empresa en concret, la qual vol beneficiar sempre que sigui possible als seus clients més fidels.
- *Clients de l'empresa client*: Aquests volen que l'empresa a la qual compren valori el seu compromís.

2. Justificació

En els darrers anys, l'empresa client ha augmentat considerablement els seus clients. Paral·lelament, el nombre de comandes també ha incrementat arribant a acumular-se i obligant a seleccionar quines es serveixen amb l'estoc existent. Aleshores, el que busca l'empresa client és tenir eines i recursos eficients per tal de beneficiar el seu rendiment i satisfer als seus clients més fidels. D'aquesta manera, a la llarga, prioritzar la clientela que més els interessa els permetrà conservar el perfil de client que consideren més important.

Al ser un projecte força diferent als que es solen desenvolupar a Seidor per a empreses concretes, serà un projecte nou, on no es poden agafar idees de solucions semblants existents, ja que el que proposa el client és força concret i no hi ha solucions que s'adeqüin del tot al que necessitem.

L'empresa client no ha establert criteris clars a l'hora de quins clients prioritzar i, per tant, s'ha considerat adient parametritzar-ho i deixar que l'empresa pugui decidir els criteris concrets de priorització.

Per tant, el nou sistema a desenvolupar ha requerit implementar un algoritme que donats uns criteris de priorització definits prèviament per l'empresa (com podrien ser, el número de comandes prèvies fetes pel client, els anys que fa que aquest és

client de l'empresa o bé els ingressos que aquest ha aportat a l'empresa), un conjunt de restriccions (com és l'estoc dels articles) i un número elevat de comandes, ens retorni el conjunt òptim de comandes a servir i l'assignació dels seus articles als diversos magatzems de l'empresa.

En el plantejament inicial del treball es van distingir dos possibles mètodes o tècniques vistes a assignatures de l'especialitat de computació del Grau en Enginyeria Informàtica de la FIB per a resoldre aquest problema. D'una banda aplicar algun algoritme d'optimització i, de l'altre, donat que és un problema exponencial i no sabíem si podríem resoldre-ho amb un temps raonable, consideraríem utilitzar una tècnica que obtingués una solució aproximada al problema d'optimització. En un inici les tècniques que es van considerar van ser la codificació del problema com un problema de *Satisfactibilitat Booleana* i utilitzar un *Sat-Solver [3]* per a trobar una solució òptima, i implementar un algoritme greedy millorat per buscar una solució aproximada. Tot i així, com es veurà més endavant, també es va analitzar, dissenyar i comparar un algoritme de *Branch&Bound* com a tècnica d'optimització.

3. Abast

3.1. Objectiu

L'objectiu principal d'aquest projecte és aportar la millor solució possible al problema que ens planteja el client, una priorització de l'assignació d'estoc disponible.

- **Paràmetres de priorització**
    - S'ha de permetre a l'empresa poder configurar de forma senzilla els paràmetres de priorització com els convingui depenent dels seus interessos.
- **Selecció del conjunt òptim de comandes**
    - El sistema ha de seleccionar el conjunt òptim de comandes a servir d'acord amb els criteris de priorització tenint en compte les restriccions d'estoc.
- **Solució coherent**

    - La solució ha de ser fàcilment entendible, coherent i esperada pels usuaris de l'empresa perquè així els sigui fàcil corregir els paràmetres de priorització adientment en cas que ho considerin necessari.

- **Distribució de magatzems**
    - Una vegada obtingut el conjunt de comandes a servir, cal distribuir els productes d'aquestes entre els diversos magatzems dels que disposa.

3.2. Subobjectius

Per aconseguir l'objectiu principal del projecte cal distingir diversos subobjectius:

- **Parametrització dels criteris de priorització**
    - Pantalla que ens permet parametritzar els criteris.
    - Incloure tots els criteris que l'empresa pugui considerar importants.
    - La importància de cada paràmetre ha de ser escalable.

- **Gestió de fitxers provinents del servei web**
    - Descomprimir i comprimir fitxers JSON.
    - Introduir comandes a l'ERP.

- **Gestió de la solució**
    - Mostrar la solució del problema de la forma més adient.

3.3. Requeriments no funcionals

A part de l'objectiu principal del projecte i els diversos subobjectius, hi ha un seguit de requeriments no funcionals que cal complir perquè l'empresa client pugui utilitzar la funcionalitat correctament.

- **Facilitat d'ús:** Els usuaris de l'empresa han de ser capaços d'utilitzar la funcionalitat i variar els criteris de priorització amb facilitat.
- **Seguretat:** Només han de poder variar els criteris de priorització els usuaris als que els està permès.

3.4. Possibles obstacles i riscos

Inicialment es van identificar els següents obstacles i riscos els quals podien dificultar aconseguir tots els objectius i requeriments del projecte:

- **Inexperiència:** Obstacles amb els que ens podíem trobar a la fase de desenvolupament del projecte a causa d'alguna inexperiència.
    - Desconeixement de SBO: Malgrat els mesos interactuant amb SAP Business One l'autor d'aquest projecte no dominava plenament l'aplicatiu.
- **Entorn local:** El fet de desenvolupar el projecte en un entorn local amb una base de dades de test per tal de no comprometre dades del client podia suposar algun obstacle a l'hora de similar l'estructura muntada en el servidor del client.
- **Errors d'implementació:** Errors amb els que ens podíem topar a la fase de desenvolupament.
    - Bugs: Errors de codi que podien suposar un obstacle, no poder avançar amb normalitat i fer perdre el temps a l'autor del projecte fins ser localitzats i solucionats.
    - Eficiència: Era important tenir en compte l'eficiència a l'hora d'obtenir una solució de l'objectiu principal.
- **Gestió del temps:** La falta de temps podia suposar un obstacle en qualsevol fase del projecte, deguda a una mala planificació o per a qualsevol altre contratemps.
- **Resultat inesperat per al client:** Podia ser que el resultat final no convencés del tot al client.
    - Incomprensió: Podia ser que els usuaris de l'empresa client no comprenguessin com utilitzar correctament la nova funcionalitat.
    - Resultat ineficaç: Podia ser que el resultat no convencés a l'empresa client.

3.5. Gestió dels riscos i obstacles

A l'apartat anterior hem identificat els possibles riscos i obstacles amb què podíem topar en la realització d'aquest treball. En el seu moment vam considerar com els podíem evitar o gestionar en cas que aquests ocorreguessin.

En quant al possible obstacle d'inexperiència amb Sap Business One, la solució que se'ns va acudir va ser dedicar més hores a la tasca prèvia identificada com a "Dominar l'estructura de comandes a SBO", tot i que l'increment d'hores tampoc hagués set molt notable, ja que a l'hora d'estimar el temps d'inversió en aquesta tasca ja s'havia tingut força en compte.

Fixant-nos amb l'obstacle d'haver de desenvolupar el projecte en un entorn local, sabiem que si calia comptaríem amb l'ajuda de treballadors de *Seidor* amb experiència, així com el tutor d'aquest treball, cosa que ens va permetre reduir la complexitat que això podia tenir.

Referint-nos als possibles errors d'implementació que podíem tenir, vam creure que no ens portaria més de 2 hores resoldre'ls i que en el cas de trobar-nos-hi simplement hi dedicaríem més hores aquell dia fins a solucionar-lo, per tant, no això no ens influïa a la planificació de les demés tasques.

Respecte al possible obstacle que ens vam plantejar d'una mala gestió del temps vam creure que ja l'haviem solventat força al haver fet prèviament una identificació de les tasques a realitzar i una bona planificació del temps invertit a cadascuna.

Per últim, vam plantejar el possible risc de que el desenvolupament fos inesperat pel client a qui va dirigit i per a solucionar aquest vam creure que ens podriem ajudar de la metodologia *agile Scrum*, ja que a l'anar mantenint el contacte i revisant el que anavem avançant amb el director del projecte seriem capaços de corregir els errors abans que aquests ens suposessin una gran pèrdua de temps. En cas d'haver-nos de desviar una mica del previst o de no tenir del tot clar com dissenyar alguna part ens hauriem de reunir amb l'empresa client, qui ens encaminaria pel camí que calia seguir. En aquest cas, les reunions ens podien suposar un augment d'un màxim de 10 hores dedicades al projecte i suposaria haver d'afegir el recurs software del "Microsoft Teams", eina utilitzada per a comunicar-nos amb l'empresa.

4. Metodologia i seguiment

Per a dur a terme el desenvolupament d'aquest projecte s'ha utilitzat la metodologia *Scrum*. Es va definir un diagrama de Gantt (el podem veure a l'apartat 5.4) amb totes les tasques que calia realitzar agrupades en sprints. Al finalitzar cada sprint es realitzava una reunió amb el director del projecte per comprovar que s'avançava d'acord amb els objectius i requeriments del projecte.
D'altra banda, es va realitzar un seguiment periòdic amb el ponent del treball per tal de verificar que es progressava correctament i d'acord amb els estàndards establerts.

Agrupar les tasques en diversos *sprints* ens va ajudar a organitzar-nos millor i saber reaccionar en els moments que ens vam endarrerir, com va ser en el cas de cercar diverses opcions per resoldre el problema d'optimització.

5. Planificació

El projecte es va iniciar a mitjans del mes de febrer i havia de finalitzar el mes de juny, per tant, es van definir i repartir les tasques del projecte entre el 14 de febrer i el 23 de juny. En la planificació inicial es va estimar que es destinarien unes 58 hores a la part de treball previ del projecte, 175 hores a la part de documentació i 260h a la part de desenvolupament, és a dir, un total de 493 hores invertides en el projecte. L'empresa client no ens havia fixat una data límit i, per tant, no va influir en el plantejament i distribució del temps.

5.1. Identificació de tasques

Es van identificar tres tipus de tasques, les de gestió, les de treball previ i les de desenvolupament. A la taula del resum de tasques (*Taula 6.1*) podem veure un resum de totes les tasques amb els temps estimats, les dependències i els recursos necessaris de cada una.

5.1.1. Tasques de gestió

La gestió del projecte agrupa totes les tasques no tècniques del projecte, les quatre primeres emmarcades a GEP (Gestió de Projectes), assignatura vinculada al treball

de final de grau. Les tasques de gestió són necessàries per planificar, definir i documentar el projecte.

TG1: Definició de l'abast i contextualització

Es dóna un context al treball, una justificació, es defineix un abast, es determinen objectius i la metodologia de treball a seguir.

TG2: Planificació temporal

Es defineixen les tasques del projecte i les possibles dependències entre elles, es fa una estimació temporal de cada una, es construeix el diagrama de Gantt i es gestionen els riscos i obstacles definits anteriorment.

TG3: Gestió econòmica i sostenibilitat

Es realitza una estimació del pressupost de cada tasca i es fa un anàlisi de la sostenibilitat del projecte.

TG4: Document final de la fase inicial (Document final de GEP)

Document on s'ajunten les tres entregues anteriors i s'apliquen els canvis que considerem necessaris basant-nos en el *feedback* proporcionat pel nostre tutor de GEP a cada tasca.

TG5: Memòria

Document principal del TFG que engloba el document final de GEP i tota la part tècnica i de desenvolupament del projecte.

TG6: Presentació

És important preparar molt bé la defensa del nostre projecte per tal que la presentació sigui fluïda i entenedora pel tribunal.

5.1.2. Tasques de treball previ

Abans de començar amb les tasques de desenvolupament calia dur a terme un treball previ per conèixer millor tot l'entorn de desenvolupament.

TTP1: Conèixer el format JSON

Buscar informació sobre com s'estructuren les dades en aquest format i sobre com es tracten.

TTP2: Dominar l'estructura de comandes a SBO

Aprofundir en el domini i coneixement del tractament de comandes a SAP Business One.

TTP3: Preparar entorn de test

Generació de comandes de test en format JSON segons els articles i clients de la base de dades de test i afegir articles, clients i/o comandes per tal de poder realitzar les proves necessàries.

TTP4: Tria de la tècnica algorítmica

Realització d'un estudi sobre la tècnica algorítmica a utilitzar per tal de resoldre l'objectiu principal del treball.

Més endavant, a l'apartat de planificació definitiva, veurem que finalment s'han destinat moltes més hores a aquesta tasca i ha passat a ser una de les més importants de tot el projecte.

5.1.3. Tasques de desenvolupament

El desenvolupament del projecte agrupa totes les tasques tècniques i funcionals del projecte, així mateix, aquestes són essencials per a obtenir un producte final que compleixi amb els objectius del projecte. Cal destacar que cada tasca inclou les proves necessàries per comprovar la seva correctesa i funcionalitat.

TD1: Introduir comandes

Tasca encarregada de fer un desenvolupament que permeti introduir les comandes en format JSON a l'ERP (Sap Business One).

TD2: Disseny i implementació de la pantalla de parametrització dels criteris de priorització

Tasca encarregada de dissenyar i implementar la pantalla on l'empresa client podrà parametritzar els criteris de priorització referents als clients que més els interessi.

TD3: Càlcul de la puntuació de cada client

Tasca encarregada de realitzar el càlcul de la puntuació que té cada client basada en els criteris establerts. Cal analitzar i realitzar varies proves sobre com dur a terme aquesta puntuació ja que es consideren varis criteris i cal adequar la manera d'extreure la puntuació de cada un d'ells.

TD4: Puntuació de cada client segons els criteris de la pantalla de parametrització

Un cop finalitzades les dues tasques anteriors es calcula la puntuació de cada client en funció dels criteris seleccionats a la pantalla de parametrització.

TD5: Càlcul de la puntuació de cada comanda

Càlcul de la puntuació de les comandes tenint en compte la puntuació del client. En el cas d'utilitzar un algoritme de *greedy* per obtenir una bona puntuació hauríem de tenir en compte altres aspectes, com per exemple, l'estoc disponible dels productes de la comanda.

Aquesta era una tasca important en el desenvolupament del projecte i calia realitzar proves per validar que les puntuacions de les comandes fossin coherents amb la priorització dels clients desitjada, donat que la solució buscada consisteix en optimitzar (o aproximar en cas que s'hagués escollit implementar un greedy) la puntuació de les comandes seleccionades.

TD6: Obtenció del conjunt de comandes a prioritzar

Selecció del conjunt òptim de comandes a servir d'acord amb les puntuacions de les comandes obtingudes en la tasca anterior.

Aquesta és la tasca principal del projecte. Consisteix en resoldre el problema d'optimització que maximitza la puntuació de les comandes seleccionades tenint en compte les restriccions d'estocs dels productes.

Calia considerar dues alternatives. En primer lloc intentar aplicar una tècnica d'optimització per obtenir la màxima puntuació i, en cas de no aconseguir un algoritme viable pel temps que requeriria trobar la millor solució, resoldre'l amb un algoritme voraç (*greedy*) que intentés aproximar-se a la millor solució.

En la planificació inicial es va considerar intentar resoldre el problema d'optimització a través d'un Sat-Solver però, com veurem a la planificació definitiva, també s'ha implementat un algoritme de *Branch and Bound* i s'ha realitzat una comparativa entre aquestes dues tècniques per decidir quina és la més adequada pel tipus de comandes de l'empresa.

TD7: Assignació de magatzems

Un cop obtinguda la llista de comandes a servir resultant de la tasca anterior, per cadascuna d'elles es realitza una assignació d'estoc dels magatzems a cadascun dels seus articles, amb la possibilitat d'assignar unitats d'un mateix article d'una comanda de magatzems diferents.

TD8: Construcció de la solució/resposta

Una vegada obtinguda la solució amb l'assignació dels estocs dels magatzems a cadascun dels productes de les comandes, es transforma la notació a format JSON per tal d'enviar la resposta al servei web.

TD9: Visualització de la solució

Per tal de poder mostrar i analitzar les respostes que proporciona el nostre sistema també es genera un fitxer excel amb la selecció de comandes i assignació dels estocs dels magatzems

5.2. Recursos necessaris

A continuació es mencionen els recursos humans i materials necessaris per poder dur a terme el projecte.

5.2.1. Recursos humans

Els recursos humans del projecte es limiten a l'autor i al director d'aquest.

L'autor s'encarrega de planificar el projecte i escriure la documentació, també és qui dissenya, analitza i desenvolupa les tècniques i algoritmes que s'utilitzen, així com també és el programador de totes les tasques de desenvolupament, i per últim, qui

realitza les proves necessàries per a comprovar que el sistema funciona correctament.

El director del projecte és l'encarregat de supervisar que totes les tasques s'adeqüin als objectius del projecte. Realitza el seguiment al finalitzar cada sprint i ajuda a redefinir els sprints quan es produeixen desviacions respecte la planificació.

5.2.2. Recursos software

A continuació es mencionen els diferents recursos software necessaris per a la realització del projecte.

SBO: SAP Business One [4] - Programa per a la gestió d'empreses amb el que es treballa a Seidor, és on s'implementa el projecte desenvolupat.

BD: Base de dades - Des de Seidor se'ns proporciona accés a una base de dades de test per tal de no comprometre dades de l'empresa client.

SQL: SQL Server [5] - Eina que farem servir per a fer les consultes necessàries a la base de dades sobre la que treballarem.

GD: Google Drive [6] - S'utilitzarà per a documentar tot el projecte, eina que ens permet compartir molt fàcilment els documents amb el ponent del treball.

C: Canva [7] - Eina utilitzada per a generar el diagrama de Gantt.

VS: Visual Studio [8] - Entorn de desenvolupament on es programa el codi de tot el desenvolupament.

5.2.3. Recursos hardware

A continuació es mencionen els diferents recursos hardware necessaris per a la realització del projecte.

P: Portàtil - Ordinador portàtil proporcionat per Seidor, eina utilitzada a totes les tasques, tant en les de gestió per a documentar el projecte com en les de desenvolupament per a programar.

5.2.4. Espai de treball

Els recursos que engloben l'espai de treball són una cadira d'escriptori [**CE**] i un escriptori [**ESC**] necessaris per a la realització de totes les tasques del projecte.

5.2.5. Consum elèctric i internet

A l'utilitzar un ordinador portàtil per a totes les tasques necessitem electricitat [**E**] per a alimentar-lo, així com internet [**I**] per tal de poder buscar informació, mantenir el contacte amb el director del projecte o connectar-nos al servidor del client.

5.3. Estimacions i Gantt

A continuació podem observar la taula de resum de les tasques detallades en l'apartat anterior, amb la planificació inicial dels recursos i temps estimat necessaris així com les dependències entre elles.

Codi	Tasca	Temps	Dependència	Recursos
T1	**Gestió del projecte**	**175 h**		
TG1	Definició de l'abast i contextualització	25 h		P, GD, CE, ESC, E, I
TG2	Planificació temporal	15 h	TG1	P, GD, C, CE, ESC, E, I
TG3	Gestió econòmica i sostenibilitat	10 h	TG2	P, GD, CE, ESC, E, I
TG4	Document final de la fase inicial	15 h	TG3	P, GD, CE, ESC, E, I
TG5	Memòria	80 h	TG4	P, GD, CE, ESC, E, I
TG6	Presentació	30 h	TG5	P, GD, CE, ESC, E, I
T2	**Treball previ**	**58 h**		
TTP1	Conèixer el format JSON	5 h		P, CE, ESC, E, I
TTP2	Dominar l'estructura de comandes a SBO	3 h		P, SBO, CE, ESC, E, I
TTP3	Preparar entorn de test	30 h	TTP1, TTP2	P, SQL, BD, SBO, CE,

				ESC, E, I
TTP4	Tria de la tècnica algorítmica	20 h		P, CE, ESC, E, I
T3	**Taques de desenvolupament**	**260 h**		
TD1	Introduir comandes	15 h	TTP1, TTP2	P, SBO, VS, BD, CE, ESC, E, I
TD2	Disseny i implementació de la pantalla de parametrització dels criteris de priorització	30 h		P, SBO, VS, SQL, BD, CE, ESC, E, I
TD3	Càlcul de la puntuació de cada client	30 h	TD1	P, SBO, VS, BD, CE, ESC, E, I
TD4	Càlcul de la puntuació de cada client segons els criteris de la pantalla de parametrització	15 h	TD2, TD3	P, SBO, VS, BD, CE, ESC, E, I
TD5	Càlcul de la puntuació de cada comanda	70 h	TD3	P, SBO, VS, BD, CE, ESC, E, I
TD6	Obtenció del conjunt de comandes a prioritzar	40 h	TTP4, TD5	P, SBO, VS, BD, CE, ESC, E, I
TD7	Assignació de magatzems	10 h	TD6	P, SBO, VS, BD, CE, ESC, E, I
TD8	Construcció de la solució/resposta	20 h	TD7	P, SBO, VS, CE, ESC, E, I
TD9	Visualització de la solució	30 h	TD8	P, CE, ESC, E, I

Taula 5.1: Taula de resum de les tasques de la planificació inicial, amb les dependències i recursos de cada una. Elaboració pròpia.

A continuació podem observar el diagrama de Gantt de les tasques proposades Tal i com s'ha mencionat anteriorment, aquestes tasques es van agrupar en diversos sprints que es mostren en el diagrama cada un amb un color diferent.

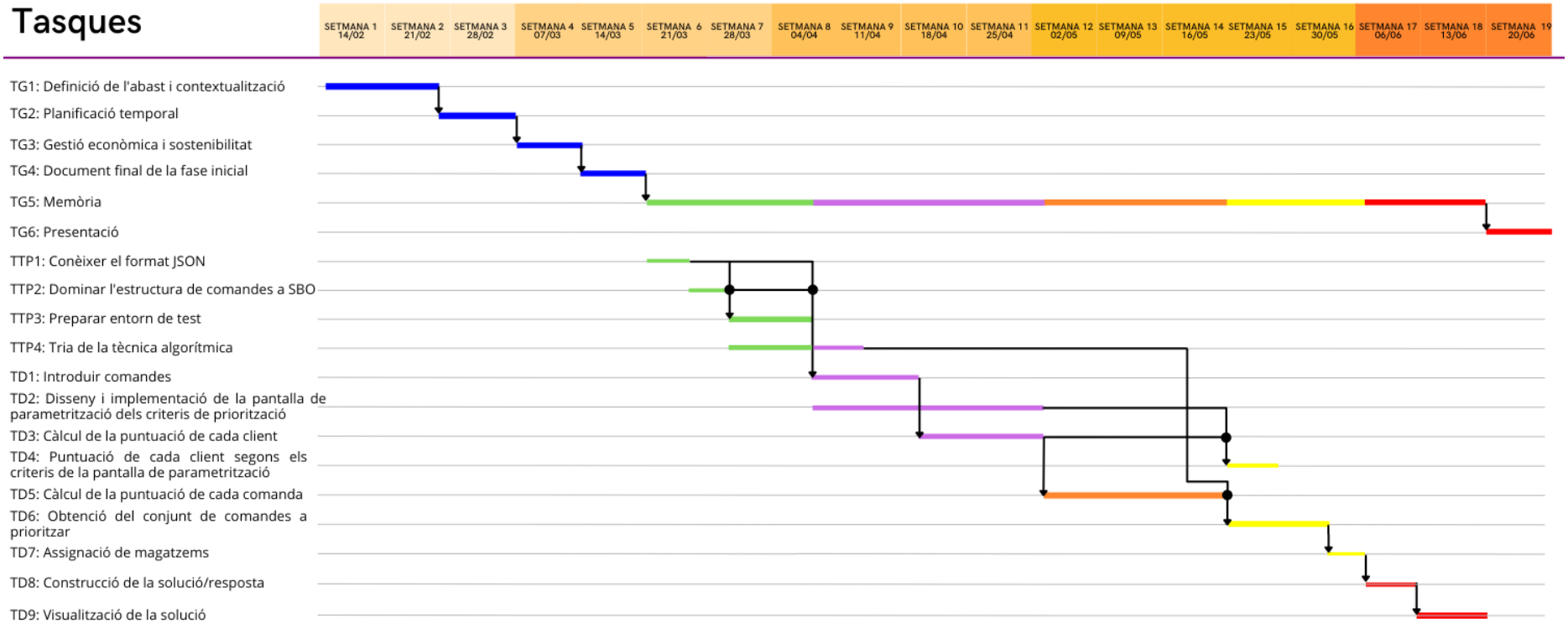


Figura 5.1: Diagrama de Gantt que representa la planificació inicial del projecte. Elaboració pròpia.

5.4. Planificació definitiva

Seguidament podem veure alguns canvis que s'han hagut de dur a terme respecte la planificació inicial d'aquest projecte. Aquests canvis no han afectat a l'objectiu final ni tampoc han suposat un gran canvi respecte els costos estimats a l'apartat de pressupost (*apartat 6*) ja que l'única variació que podem trobar és un augment o disminució dels costos de personal a l'haver de dedicar més o menys hores en algunes tasques, però les hores totals dedicades al projecte s'acaben ajustant força a les que ja vam estimar inicialment.

El primer canvi el qual ens va fer endarrerir algunes de les tasques planificades inicialment va estar haver d'afegir una nova tasca de treball previ, la instal·lació dels web services de SAP (en podem veure el detall al següent subapartat).

D'altra banda, el temps invertit a la tasca dedicada a la tria de la tècnica algorítmica (*tasca TTP4*) per a resoldre el problema de la priorització de l'estoc ha acabat sent molt superior al que a priori vam estimar. Tanmateix, no es tracta d'una tasca prèvia tal com vam identificar sinó que en aquesta planificació definitiva la identifiquem com a una tasca de desenvolupament en la qual aprofundim molt més i on també s'inclouen una sèrie de proves entre els algoritmes analitzant els seus comportaments.

Per últim, un canvi poc notori sobre l'objectiu del projecte ha estat suprimir la tasca orientada a construir la resposta per a enviar-la al servei web (*tasca TD8*), ja que l'única resposta que retorna al servei web és si la comanda s'ha introduït correctament a SAP o no, però en cap circumstància s'envia al servei web l'assignació de l'estoc. Aquest canvi en la planificació l'ha causat una confusió amb el client a l'hora de planificar el desenvolupament.

5.4.1. Tasques afegides

TTP5: Instal·lació dels Web Services de SAP

A l'apartat de justificació del projecte ja vam comentar que les comandes que arriben al client provenen d'un servei web, per tant, per a adaptar el problema del client a l'entorn local on es desenvolupa aquest projecte, cosa que ja havíem

esmentat a l'apartat dels possibles obstacles amb els quals podíem topar, vam necessitar instal·lar els Web Services que proporciona SAP en el servidor web de windows *IIS (Internet Information Services)* de la màquina local els quals ens permeten simular la introducció de comandes provinents d'un servei web.

D'aquesta manera, a l'hora de passar el desenvolupament fet a la màquina local al servidor del client només s'haurà de canviar la referència del servei web.

5.4.2. Recursos afegits

IIS: Internet Information Services [9] - Servidor web de Windows on s'instal·len els Web Services que necessitem. Recurs utilitzat a la nova tasca identificada al subapartat anterior d'instal·lació dels Web Services de SAP (*TTP5*).

Z3: SMT Solver [10] - Solucionador multiplataforma de problemes *SMT* (*Satisfiability Modulo Theories*) de Microsoft.

5.4.3. Diagrama de Gantt definitiu

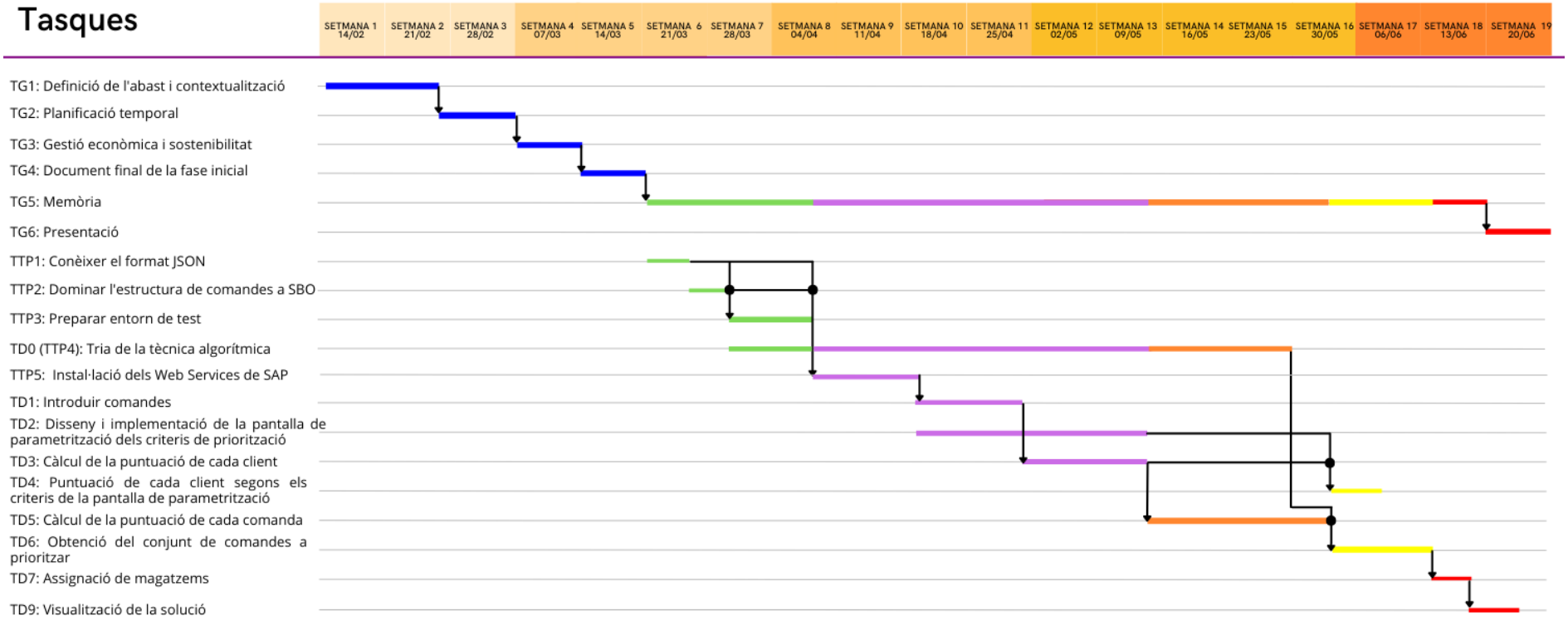


Figura 5.2: Diagrama de Gantt que representa la planificació definitiva del projecte. Elaboració pròpia.

6. Pressupost

6.1. Identificació i estimació dels costos

Per a determinar la viabilitat econòmica del projecte va ser necessari identificar els costos que suposava. Calia tenir en compte els costos de personal, els costos genèrics, les amortitzacions, un pla de contingència i els costos per possibles imprevistos.

6.1.1. Costos de personal

En aquest projecte podem distingir entre dos perfils o rols de personal, per una banda, el rol de cap de projecte (compartit entre director i autor del treball) i per altra banda, el rol de desenvolupador (dut a terme únicament per l'autor del treball). A la següent taula (*Taula 8.1*) podem observar els costos estimats en euros/hora que corresponen a cada perfil. El sou brut de cada rol l'hem extret de la calculadora salarial de *Hays* [11] la qual ens dóna un sou aproximat anual per càrrec. Per a calcular el sou brut per hora suposem 160 hores treballades cada mes i per a calcular el cost total per hora multipliquem aquest per 1.3 per tal d'incloure l'aportació establerta per a la seguretat social.

Rol	**Sou brut anual**	**Sou brut per hora**	**Cost per hora (tenint en compte cost SS)**
Cap de projecte (C)	55.000 €/any	28,65 €/h	37,25 €/h
Documentador (D)	25.000 €/any	13,02 €/h	16,93 €/h
Programador (P)	30.000 €/any	15,63 €/h	20,32 €/h

Taula 6.1: Taula amb els costos estimats per rol en euros hora. Elaboració pròpia.

Una vegada definits els costos de personal per hora i rol ja podem estimar el cost de cada tasca de la planificació definitiva vista al diagrama de Gantt de l'anterior subapartat. A la següent taula podem observar els costos de personal que suposa cada tasca així com el cost de personal total d'aquest projecte.

Codi	Tasca	Cap de projecte	Documentador	Programador	Cost
T1	**Gestió del projecte**	**40 h**	**135 h**		**3.775,7 €**
TG1	Definició de l'abast i contextualització	10 h	15 h		626,5 €
TG2	Planificació temporal	5 h	10 h		355,6 €
TG3	Gestió econòmica i sostenibilitat	5 h	5 h		270,9 €
TG4	Document final de la fase inicial		15 h		254 €
TG5	Memòria	20 h	60 h		1760,8 €
TG6	Presentació		30 h		507,9 €
T2	**Treball previ**	**10 h**		**48 h**	**975,4 €**
TTP1	Conèixer el format JSON			5 h	101,6 €
TTP2	Dominar l'estructura de comandes a SBO			3 h	61 €
TTP3	Preparar entorn de test			30 h	609,6 €
TTP5	Instalació dels Web Services de SAP	10 h		10 h	203,2 €
T3	**Taques de desenvolupament**			**290 h**	**6.096 €**
TD0 (TTP4)	Tria de la tècnica algorítmica			100 h	2.032 €
TD1	Introduir comandes			15 h	304,8 €
TD2	Disseny i implementació de la pantalla de parametrització dels criteris de priorització			30 h	609,6 €
TD3	Càlcul de la puntuació de cada client			30 h	609,6 €
TD4	Càlcul de la puntuació de cada client segons els criteris de la pantalla de parametrització			15 h	304,8 €
TD5	Càlcul de la puntuació de cada comanda			30 h	609,6 €
TD6	Obtenció del conjunt de comandes a prioritzar			30 h	609,6 €
TD7	Assignació de magatzems			20 h	406,4 €
TD9	Visualització de la solució			20 h	609,6 €

Total	**50 h**	**135 h**	**338 h**	**10.847,1 €**

Taula 6.2: Estimació dels costos de personal segons les hores de dedicació de cada rol a les tasques de la planificació definitiva. Elaboració pròpia.

Cal destacar que, en el cas de necessitar adaptar aquesta solució per a altres clients, es reduirien notòriament les hores destinades a la tria de la tècnica algorítmica i consegüentment els costos de personal.

6.1.2. Costos genèrics i amortitzacions

Els costos genèrics són aquells generalment calculats a partir dels recursos utilitzats durant el desenvolupament del projecte (mencionats anteriorment a l'apartat 5.3), com que el projecte tenia una durada de 4 mesos, les amortitzacions dels recursos es van calcular entorn els quatre mesos. Dintre d'aquests podem distingir el *software* i *hardware*, l'espai de treball, el consum elèctric i l'internet.

6.1.2.1. Software

Tot el programari utilitzat en aquest projecte és gratuït a excepció de la llicència professional de l'ERP SAP Business One, la qual té un cost de 2700€ l'any (12 mesos). Per tant, l'amortització d'aquest recurs és de 2700/12*4 = **900 €**.

6.1.2.2. Hardware

L'únic recurs *hardware* utilitzat per a la realització d'aquest projecte és un ordinador portàtil "*Lenovo Thinkpad T450*" el qual té un cost de 1269,23€. Suposant que el temps de vida útil d'aquest ordinador portàtil és d'uns 5 anys (60 mesos) l'amortització d'aquest recurs és de 1269,23/60*4 = **84,62 €**.

6.1.2.3. Espai de treball

Els recursos materials utilitzats al nostre espai de treball es limiten a un petit estudi/habitació d'uns 12 m^2 amb una cadira d'escriptori [12] de cost 209,90€ i un escriptori [13] de cost 79€.

Sabent que el lloguer del pis on es troba l'estudi té un cost de 500€ mensuals i un espai total d'uns 120 m^2, s'amortitzen 12*500/120 = 50€ del lloguer del pis cada mes, per tant, una amortització total de 50*4 = 200€ d'aquest lloguer. Suposant que

el temps de vida útil tant de la cadira com de l'escriptori és d'uns 10 anys (120 mesos), s'amortitzen 209,9/120*4 = 7€ de la cadira i 79/120*4 = 2,63€ de l'escriptori. Per tant, l'amortització total de l'espai de treball que utilitzem ascendeix als **209,63 €**.

6.1.2.4. Consum elèctric

Per a estimar el cost de l'electricitat consumida al llarg del projecte calia saber el preu de l'electricitat, el qual varia en funció del dia i l'hora, per tant, es va prendre com a referència la mitjana del cost de l'electricitat del dia en que es va realitzar l'estimació, el qual estava a 0,28458€/kWh [14]. Tenint en compte que l'únic recurs utilitzat que consumeix electricitat és l'ordinador portàtil, el qual té una potència d'uns 100W i s'utilitza en totes les tasques (500h aprox), s'estima una despesa de 50kWh els quals suposen un cost de **14,23 €**. Amb els increments del preu de l'electricitat en els següents mesos aquest cost ha incrementat, però malgrat tot, el cost del consum elèctric és molt menor en aquest projecte.

6.1.2.5. Internet

La factura mensual d'internet a casa puja a uns 50€. Es va estimar una dedicació d'unes 500 hores al projecte que tenia una durada d'uns 4 mesos, per tant, 125 hores mensuals en les que s'utilitzaria internet. Per tant, es va estimar una amortització total de l'internet d'uns 125/(30*24) * 50 * 4 = **34,72 €**.

6.1.2.6. Costos genèrics totals

A la següent taula hi trobem un resum de l'estimació dels costos genèrics del projecte, així com el total d'aquests.

Recurs	**Cost**
Software	900 €
Hardware	84,62 €
Espai de treball	209,63 €
Consum elèctric	14,23 €
Internet	34,72 €

Total de costos genèrics	**1.243,20 €**

Taula 6.3: Resum dels costos genèrics estimats del projecte. Elaboració pròpia.

6.1.3. Contingències

Durant el transcurs del projecte poden aparèixer diversos contratemps i complicacions, per afrontar-les es va preparar una partida de contingència. Aquesta partida de contingència es tradueix en un cost que es calcula sumant els costos de personal i genèrics que estimats i multiplicant-los pel nivell de contingència. Per tant, assumint un nivell de contingència del 15% el cost de contingència és de (10.847,1 + 1243,20)*0,15 = **1.813,55 €**.

6.1.4. Imprevistos

Durant la realització de qualsevol projecte poden sorgir imprevistos i per a cada un d'aquests és necessari plantejar-se un pla alternatiu per si en algun moment s'ha d'afrontar. El cost que suposa cada imprevist es calcula com la probabilitat que aquest ocorri multiplicat per la despesa que suposaria el seu pla alternatiu. A la següent taula podem observar els costos estimats que suposarien els imprevistos amb els que podem topar:

Imprevist	Temps i rol	Despesa	Probabilitat	Cost
Falta de temps en la part de gestió	20h C	745 €	20 %	149 €
Falta de temps en la part de documentació	20h D	338,6 €	20 %	67,7 €
Falta de temps en la part de desenvolupament	30h P	609,6 €	20 %	121,9 €
Bugs	30h P	609,6 €	40 %	243,8 €
Resultat inesperat pel client	40h P	812,8 €	10 %	81,3 €
Cost total per a imprevistos				**663,7 €**

Taula 6.4: Estimació dels costos causats per imprevistos. Elaboració pròpia.

6.1.5. Cost total

A la *taula 8.5* podem veure un resum del pressupost teòric final d'aquest projecte, on no apareixen les contingències ni els imprevistos donat que ja s'han ajustat les hores de la planificació definitiva (reals) als costos de personal.

Concepte	**Cost**
Costos de personal	10406,80 €
Costos genèrics i amortitzacions	1243,20 €
Cost total del projecte	**13882,2 €**

Taula 6.5: Pressupost teòric final del projecte. Elaboració pròpia.

7. Informe de sostenibilitat

7.1. Autoavaluació

Després de respondre el qüestionari genèric per a estudiants d'enginyeria [15] l'objectiu del qual es basa en obtenir informació sobre els nostres coneixements en relació amb els Objectius de Desenvolupament Sostenible (ODS) i les competències en sostenibilitat on s'inclouen les dimensions social, ambiental i econòmica considerem que el nostre nivell de coneixement sobre l'impacte que pot tenir en termes de sostenibilitat un projecte de gran dimensió com és aquest no és gaire elevat, ja que els projectes que hem realitzat fins ara han estat molt més breus i no ha fet falta fixar-nos-hi massa. Dur a terme aquest qüestionari ens ha ajudat a veure els punts que menys dominem i amb els que més ens haurem de fixar a l'hora de posar-los en pràctica.

Cal mencionar que aquest projecte no fa ús de cap matèria primera i l'únic requisit *hardware* imprescindible és un ordinador portàtil, per tant, les decisions que puguem prendre per a reduir sobretot l'impacte ambiental seran força limitades.

Respecte als impactes econòmic i social, sí que podríem influir-hi més intentant rendibilitzar al màxim les hores invertides al projecte, les quals suposen una notable

despesa en termes econòmics i intentant satisfer al màxim el que busca l'empresa client, millorant així, la societat que engloba l'empresa i clients.

7.2. Dimensió ambiental

En el desenvolupament del projecte només es fa ús material d'un ordinador portàtil de segona mà proporcionat per l'empresa Seidor, per tant, l'impacte ambiental no és significatiu, ans al contrari, permet minimitzar-lo donant un segon ús a l'ordinador i reduint el consum energètic en cas que s'utilitzés un ordinador de torre.

Respecte al resultat del projecte -la funcionalitat que s'afegirà-, aquest no suposa cap canvi en la dimensió ambiental, ja que no s'utilitzaven ni s'utilitzarà cap material que pugui ser perjudicial pel medi ambient.

7.3. Dimensió econòmica

A l'apartat 6 s'han estimat el cost de recursos humans que ha suposat aquest projecte partint d'una aproximació del que solen pagar les empreses perquè des de Seidor els desenvolupin noves funcionalitats dintre de l'entorn de l'ERP SAP Business One i l'amortització dels materials utilitzats durant el transcurs del projecte.

Referint-nos a la vida útil del projecte, la nova funcionalitat que proporciona la nostra solució pot arribar a suposar una notable millora econòmica per a l'empresa, ja que permetrà beneficiar els clients que aquesta considera més importants, i això, en termes econòmics es pot reflectir a un augment de les comandes per part dels clients que suposen més ingressos a l'empresa.

7.4. Dimensió social

La realització d'aquest projecte ens ha aportat, en l'àmbit de gestió, noves maneres de planificar i organitzar tasques, i en l'àmbit tècnic, coneixements sobre l'estructura dels documents de venda a l'ERP SAP Business One i habilitats per a desenvolupar i integrar noves funcionalitats a SBO. Tanmateix, també ens ha permès aprofundir en els possibles algoritmes que permeten resoldre problemes d'optimització.

Aquest desenvolupament suposarà un ampli avantatge tant per l'empresa com pels clients que aquesta consideri més importants, els quals sortiran beneficiats i augmentaran el seu compromís amb l'empresa.

Com s'ha citat anteriorment en el primer apartat, l'empresa creu necessari afegir aquesta funcionalitat, ja que el flux de comandes que rep ha augmentat significativament en els darrers anys i en cas d'haver de prioritzar un client o un altre, a causa de no tenir suficient estoc per a repartir, el volen assignar al que més els convingui.

8. Identificació de lleis i regulacions

La única regulació que podem trobar relacionada amb aquest projecte és la *Llei Orgànica de Protecció de Dades de Caràcter Personal LOPD [22]*, una llei orgànica amb l'objectiu de protegir la intimitat personal. És per això que, per tal d'evitar un possible incompliment de la llei i com ja vam mencionar anteriorment, aquest projecte l'hem desenvolupat en un entorn local utilitzant una base de dades de test amb dades fictícies.

9. Desenvolupament del projecte

9.1. Introducció de comandes

Com ja vam comentar anteriorment, a l'apartat de planificació definitiva del projecte, per introduir les comandes abans hem d'instal·lar els Web Services de SAP. Concretament, els dos serveis de SAP que necessitem són el *LoginService*, el qual ens serveix per a identificar-nos i poder accedir tant al SAP com a la base de dades i l'*OrdersService* el qual un cop connectats amb les credencials correctes via LoginService ens permet introduir comandes.

Per tant, el primer que necessitem és afegir les referències d'aquests dos serveis al projecte de *Visual Studio* que hem creat per a introduir les comandes a SAP i un cop afegides, amb l'ajuda de l'eina *Windows Comunication Foundation (WCF)* que

aquest inclou ja podem començar a testejar la introducció de comandes a SAP simulant que aquestes provenen d'un servei web.

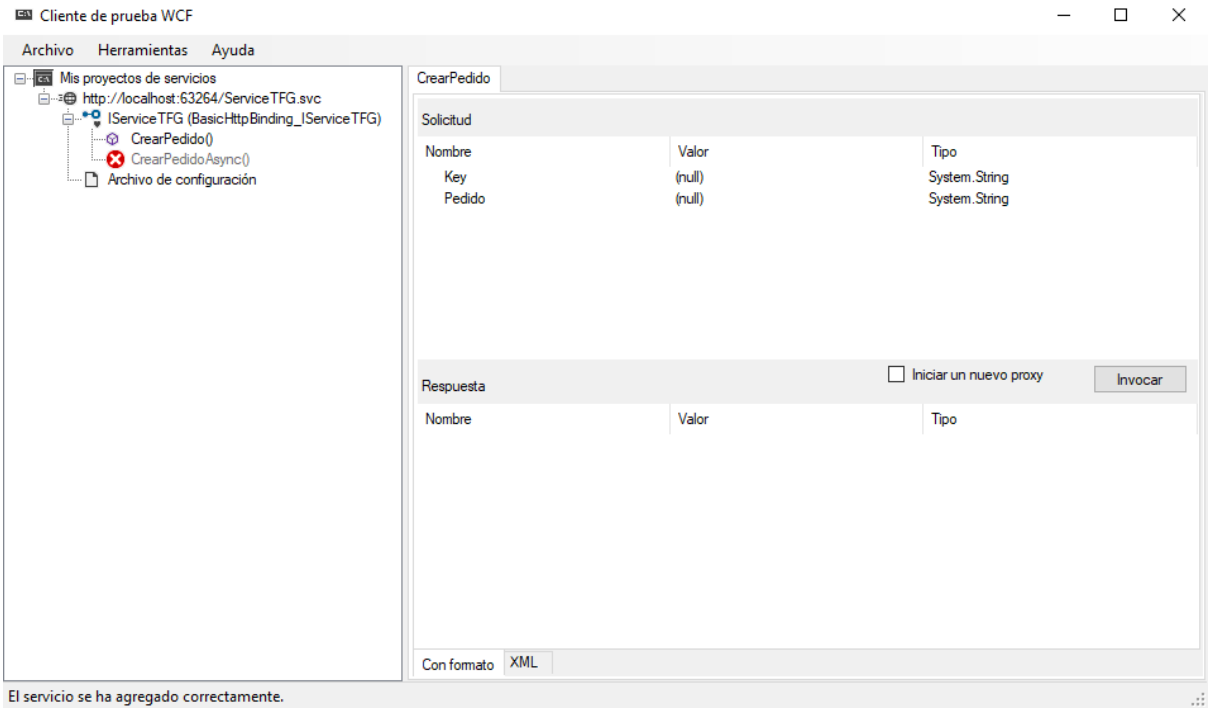


Figura 9.1: Client de prova WCF per a testejar Web Services. Elaboració pròpia

9.1.1. Estructura de les comandes

Hem afegit dos camps d'usuari a la taula *ORDR*, la qual emmagatzema les comandes de venda de SAP Business One. El primer (*U_SEIEstocAsg*), de mida 1 i tipus *alfanumèric* ens indica si la comanda ja ha passat pel procés d'assignació d'estoc i estarà cobert amb una 'S' si és que sí o amb una 'N' si és que no. El segon (*U_SEIDocHour*), de tipus *Data* i subtipus *Hora* ens indica l'hora d'arribada de la comanda, és necessari afegir-lo ja que SAP per defecte només té un camp per a guardar el dia, però no la hora.

A través del servei web del client ens arribaran les comandes amb l'estructura següent on podem distingir entre els camps de capçalera i els camps de les línies que es corresponen a cada producte de la comanda:

```
{"id_customer":"CodiClient","id_address_delivery":"DireccióEntrega","id_address_invoice":"DireccióFactura","DocDate":"DataComanda",
"Lineas":[{"product_quantity":QuantitatProducte1,"product_price":PreuProducte1,"product_reference":"CodiProducte1"},
          {"product_quantity":QuantitatProducte2,"product_price":PreuProducte2,"product_reference":"CodiProducte2"}]
}
```

Figura 9.2: Estructura comandes provinents del Web Service. Elaboració pròpia

Els camps de la capçalera i totes les línies es mapejen de manera adient a l'*OrdersService* com podem veure a la *taula 9.1* a partir del fitxer que arriba del servei web i s'introdueix la comanda a SAP. A més, quan es crea la comanda s'assignen als dos camps d'usuari *U_SEIEstocAsg* i *U_SEIDocHour* els valors *'N'* i el temps actual (en hores i minuts) respectivament.

Fitxer provinent del servei web	**Atribut de l'OrdersService**
id_customer	*CardCode*
id_address_delivery	*Address2*
id_address_invoice	*Address*
DocDate	*DocDate*
Fitxer provinent del servei web	**Atribut de l'OrdersService.Lines**
product_reference	*ItemCode*
product_quantity	*Quantity*
product_price	*UnitPrice*

Taula 9.1: Mapeig dels camps del fitxer provinent del servei web a l'OrdersService. Elaboració pròpia

9.2. Disseny i implementació de la pantalla de priorització de l'estoc

A l'hora de dissenyar aquesta pantalla s'han analitzat diverses alternatives per tal que l'usuari de forma senzilla i visual pugui fixar uns criteris de priorització i executar el procés de priorització de l'estoc.

Una opció que hem contemplat per a fixar els criteris ha estat posar davant de cada criteri una casella editable on introduir una puntuació de l'1 al 10 per tal que l'usuari pugui fixar l'importància de cada un d'aquests.

Una altra opció contemplada ha estat posar davant de cada criteri un seguit de caselles de selecció on controlem per codi que només se'n pugui marcar una per línia i en aquest cas la casella seleccionada de cada línia és la que marca l'importància del criteri en qüestió. Aquesta opció és la que més s'ajusta al que

interessa a l'usuari ja que és molt més visual i entenedora que l'anterior. A la *figura 9.3* podem observar el disseny de la pantalla.

Figura 9.3: Pantalla de parametrització dels criteris de priorització. Elaboració pròpia

El disseny de la pantalla segueix l'estil de les pantalles estàndard de *SAP Business One* ja que des de SAP consideren fonamental que totes les pantalles tinguin aquest mateix estil per tal de conservar el *look and feel* (aparença del lloc i el que sent l'usuari quan hi interactua), per tant, marquen unes directrius per a tots els seus *partners*, entre ells *Seidor*, els quals han de seguir aquest mateix patró de disseny.

A la part inferior de la pantalla hem afegit 3 botons, començant per l'esquerra trobem el botó *Prioritzar*, el qual ens permet executar el procés de priorització de les comandes pendents de prioritzar un cop fixats els criteris, al seu costat hi veiem el de *Cancelar*, botó estàndard de SAP que el que fa és senzillament tancar la pantalla tal com ho fa la creueta de dalt a la dreta sense guardar cap canvi i per últim tenim el botó *Guardar criteris*, el qual ens permet guardar els criteris de priorització indicats.

9.3. Obtenció del conjunt de comandes a prioritzar

Per a obtenir el conjunt de comandes a prioritzar abans és necessari obtenir les puntuacions dels clients i de les comandes a partir dels criteris fixats a la pantalla vista a l'apartat anterior (*figura 9.3*), seguidament, un cop obtingudes aquestes puntuacions els algoritmes dissenyats són els que s'encarreguen d'obtenir el conjunt de comandes a prioritzar.

9.3.1. Càlcul de la puntuació de cada client

Per a calcular la puntuació d'un client hem de tenir en compte els criteris que tenen a veure amb el client, és a dir, el número de compres del client, la despesa total del client i l'antiguitat d'aquest. La puntuació del client es calcula com una suma de les puntuacions dels tres criteris esmentats, on la màxima puntuació que pot obtenir per a cada un ve marcada pels nivells d'importància prèviament definits.

Després d'analitzar varies formes de calcular la puntuació que obté cada client, hem decidit que la millor opció és fixar un llindar modificable per l'usuari/empresa el qual ens indica a quin valor es troba la puntuació màxima de cada criteri. És a dir, el client obtindrà la puntuació màxima d'aquell criteri si iguala o supera aquest llindar marcat, i sinó, obtindrà la puntuació proporcional sabent que la màxima es correspon amb el llindar.

Aquests llindars els hem introduït a la taula *@SEICONTA (Contadors)*, la qual des de *Seidor* utilitzem sovint per a guardar parametritzacions de tipus (codi, nom, valor). A la següent captura podem veure els que afecten a la puntuació del client:

Code	Name	Valor
LlindarAntig	Llindar antiguitat del client	20
LlindarDespesa	Llindar despesa total del client	1.000.000
LlindarNCompr	Llindar número de compres del client	100

Figura 9.4: Registres de la taula @SEICONTA referents als clients. Elaboració pròpia

9.3.2. Càlcul de la puntuació de cada comanda

Per a calcular la puntuació de cada comanda hem de tenir en compte varies coses, per una banda els criteris de priorització fixats per l'usuari que tenen a veure amb la

comanda (cost de la comanda, data/hora de la comanda) i per l'altra la puntuació del client de la comanda, calculada prèviament com hem vist a l'apartat anterior. La puntuació que obté cada comanda de cada criteri es calcula de la mateixa forma que la puntuació dels clients. Per tant, hem afegit els següents llindars a la taula *@SEICONTA*.

LlindarCostCom	Llindar cost de la comanda	20.000
LlindarHArribCom	Llindar hores des d'arribada comanda	24

Figura 9.5: Registres de la taula @SEICONTA referents a les comandes. Elaboració pròpia

9.3.3. Disseny de l'algoritme

Com ja vam identificar prèviament a l'apartat d'identificació de tasques de la planificació d'aquest treball, el primer pas a l'hora de dissenyar l'algoritme per a resoldre l'objectiu principal és escollir la tècnica algorítmica que farem servir (*tasca TTP4*).

Mentre que en els apartats *9.1 (Introducció de les comandes a SBO via Web Service*) i *9.2 (Disseny i implementació de la pantalla de priorització de l'estoc)* ens hem ajudat dels coneixements adquirits durant el transcurs de les pràctiques extracurriculars i curriculars cursades a l'empresa *Seidor*, en aquest apartat és on entren els coneixements adquirits al llarg dels estudis de grau, sobretot enfocats en l'especialitat de computació. Concretament, en l'assignatura d'algorísmia vam veure varies tècniques algorítmiques per a solucionar problemes d'optimització (com *greedy*, *backtracking* i *Branch&Bound*).

A l'especialitat de computació també vam veure alguns problemes de satisfactibilitat booleana i d'aquí és d'on va venir la idea inicial de resoldre aquest tipus de problema amb un *SAT-Solver*. Tot i això al tractar-se d'un problema amb aritmètica entera hem hagut de buscar informació sobre els SMT Solvers i la codificació que aquests fan servir per a obtenir solucions òptimes.

9.3.3.1. Identificació del problema a resoldre

El problema a resoldre es basa en distribuir l'estoc disponible dels articles que apareixen al conjunt de comandes d'entrada per tal de cobrir un subconjunt òptim de comandes. Aquesta optimitat vindrà marcada per la puntuació que tingui cada

comanda, calculada a partir dels criteris fixats anteriorment referents a la pròpia comanda i al client d'aquesta.

Podem trobar una similitud entre el nostre problema i el problema de la motxilla [16] (conegut com *Knapsack problem*), vist a l'assignatura d'algorísmia. Aquest problema d'optimització consisteix en omplir una motxilla amb objectes que tenen un valor i pes determinats sense superar el pes màxim que aquesta suporta i maximitzant el valor total dels objectes inclosos. En el nostre cas, els objectes són les comandes, el valor és la puntuació obtinguda de cada comanda a partir dels criteris de priorització i la limitació (pes) l'estoc disponible dels articles de les comandes. Tot i això, la principal diferència que trobem entre ambdós és que mentre amb el nostre problema tenim restriccions per cada article fixades per l'estoc disponible d'aquest, amb l'altre només tenim la restricció de no superar el pes màxim que suporta la motxilla, el qual seria el mateix que el nostre problema amb només un article i la seva restricció d'estoc, per tant, al tenir varis articles que poden aparèixer a les línies de les comandes d'entrada tindrem tantes restriccions a tenir en compte com articles.

9.3.3.2. Preprocés

Independentment de la tècnica que utilitzem per a resoldre el problema, prèviament fem un preprocés el qual ens pot eliminar algunes de les comandes a tractar i divideix el problema en varis per tal de disminuir el temps d'execució de l'algoritme que després s'aplica.

En primer lloc s'eliminen les comandes que no es poden servir degut a la insuficiència d'estoc d'algun dels seus productes, a continuació s'afegeixen directament a la solució les comandes que ja es poden servir a l'haver-hi suficient estoc de tots els productes que conté considerant totes les comandes, i per últim, es separen les comandes en particions independents a nivell d'articles de manera que qualsevol comanda d'una partició no té en comú cap article amb una comanda d'una altra. Seguidament podem veure el segment de codi encarregat de fer aquestes particions.

```vb
Private Sub ParticionarComandes()
    Dim lComandes As List(Of Integer) = New List(Of Integer)
    Dim comanda As Integer
    For Each comanda In Me.dOrders.Keys
        lComandes.Add(comanda)
    Next
    While lComandes.Count > 0
        comanda = lComandes(0)
        lComandes.RemoveAt(0)
        Me.lPartitions.Add(GetParticio(comanda, lComandes))
    End While
End Sub

1 referencia
Private Function GetParticio(ByVal comanda As Integer, ByRef lComandesToCheck As List(Of Integer)) As List(Of Integer)
    Dim lParticio As List(Of Integer) = New List(Of Integer)
    Dim lProductes As List(Of String) = New List(Of String)
    Dim lProductes0 As List(Of String)
    Dim comandesIncloses As List(Of Integer)
    Dim producte As String
    Dim newComanda As Boolean
    For Each producte In Me.dLinOrders(comanda).Keys
        lProductes.Add(producte)
    Next
    lParticio.Add(comanda)
    newComanda = True
    While newComanda And lComandesToCheck.Count > 0
        newComanda = False
        comandesIncloses = New List(Of Integer)
        For Each com As Integer In lComandesToCheck
            lProductes0 = New List(Of String)
            For Each producte In Me.dLinOrders(com).Keys
                lProductes0.Add(producte)
            Next
            If Contains(lProductes, lProductes0) Then
                newComanda = True
                lParticio.Add(com)
                comandesIncloses.Add(com)
                For Each p As String In lProductes0
                    If Not lProductes.Contains(p) Then
                        lProductes.Add(p)
                    End If
                Next
            End If
        Next
        For Each com As Integer In comandesIncloses
            lComandesToCheck.Remove(com)
        Next
    End While

    Return lParticio
End Function
```

Figura 9.6: Segment de codi en VB.NET encarregat de separar les comandes en particions independents. Elaboració pròpia

9.3.3.3. Anàlisi d'alternatives per a resoldre el problema

Com ja hem mencionat abans, el nostre problema consisteix en trobar la millor solució entre totes les combinacions vàlides (respectant les restriccions de l'estoc disponible de cada article) de comandes a servir. S'entén com a millor solució la que maximitza la suma de les puntuacions de les comandes seleccionades. Enumerar totes les combinacions i buscar la millor entre les combinacions vàlides explorant-les una a una, no és una opció possible donat que requeriria explorar un nombre de combinacions exponencial respecte al nombre de comandes. És per això que

necessitem realitzar un anàlisi de les varies formes de resoldre el problema i veure quins avantatges i inconvenients trobem en cada cas.

A l'apartat de justificació del projecte només vam mencionar dos possibles tècniques per a resoldre el nostre problema, la primera dissenyant un algoritme voraç (*greedy millorat*) que retorni una solució raonable però que no ens pot assegurar obtenir una solució òptima i la segona traduir el nostre problema a clàusules entendibles per un *SAT-Solver* i que aquest ens obtingui la solució òptima. Dins d'aquesta identificació de possibles tècniques no vam incloure *Least-Cost Branch&Bound* ja que a priori no vam considerar viable dissenyar un algoritme que obtingués una solució òptima al nostre problema per qüestions d'eficiència donat que en el pitjor cas el problema té un cost exponencial, i crèiem que difícilment s'aconseguirien resultats millors o similars als que es podrien tenir utilitzant un SAT-Solver.

Tot i així, es va decidir reconsiderar-ho per dues raons. Podíem dissenyar un *Least-Cost Branch&Bound* que obtingués ràpidament una primera solució que coincidiria o milloraria la que s'obtindria amb l'algoritme voraç. Per tant, sempre tindríem l'opció d'establir un temps límit i donar la millor solució trobada fins al moment podent donar una millor aproximació. D'altra banda també s'ha volgut observar com funcionaria amb el tipus de comandes i estocs de l'empresa concreta, i també comparar el comportament respecte al resultat obtingut si el resolem a través d'un *SAT-Solver*.

Algoritme voraç

Greedy [17] (algoritme voraç) és una tècnica algorítmica que permet resoldre problemes d'optimització la qual segueix una heurística que consisteix en seleccionar l'opció òptima a cada pas de l'algoritme sense considerar futures conseqüències amb l'esperança d'arribar a un solució òptima. Consegüentment, la clau d'aquest tipus d'algoritmes rau en seleccionar el millor candidat a cada pas.

La principal avantatge que presenta aquest algoritme respecte als altres plantejats és l'eficiència en temps ja que aquest es limitaria a cada pas eliminar del conjunt de comandes candidates aquelles no factibles segons l'estoc disponible dels articles que apareixen en aquesta, seleccionar la comanda amb més puntuació, i actualitzar

l'estoc disponible dels articles que apareixen a la comanda seleccionada. Aquest procés es repetiria fins que no quedessin comandes factibles.

Per contra, l'inconvenient que té aquest algoritme és el fet de no poder assegurar que trobi una solució òptima ja que, com hem mencionat abans, un pas òptim local no ens condueix sempre cap a una solució òptima del nostre problema.

Least-Cost Branch&Bound

El *backtracking* és un algoritme general que permet trobar totes les solucions, i per tant la millor, construint gradualment els possibles candidats a solució seguint una estratègia en profunditat (*deep-first*) i descarta un candidat quan determina que no es pot convertir en una solució vàlida. Això requereix explorar tots els nodes que contenen una solució parcial o total vàlida que, en general, també té un cost exponencial.

Una de les tècniques que sovint s'utilitzen per tractar problemes d'optimització combinatòria és el mètode de *Branch and Bound [18]*. Aquest segueix una estratègia molt semblant a la del *backtracking*, però amb la diferència que no es limita a recórrer l'arbre seguint una estratègia concreta sinó que selecciona el següent node a tractar basant-se en una funció heurística. D'altra banda, es guarda informació en els nodes que permet descartar-los tan aviat com es pot saber que la millor solució que es podria obtenir a partir d'aquest no pot superar la millor solució actual.

Concretament el *Least-Cost Branch&Bound* converteix el problema d'optimització en un problema de minimització. La selecció del següent node a tractar es basa en una funció de cost heurística i es selecciona el node amb el menor cost màxim. La informació de *costMínim* en cada node, ens permet podar aquells nodes amb *costMínim* superior al cost de la millor solució trobada fins al moment.

El nostre problema, com ja hem mencionat varies vegades, tracta de maximitzar la puntuació del conjunt de comandes seleccionades. Per a transformar-lo en un problema de minimització s'ha de donar un cost màxim igual a la puntuació de totes les comandes sumades a un node inicial (*dummy*) i per a cada comanda afegida a la solució restar la puntuació d'aquesta al cost del node. Per tant, en el nostre cas,

cada node tindrà com a cost màxim la suma de les puntuacions de les comandes no seleccionades més les de les comandes que queden per considerar i, per tant, la solució òptima es trobarà al node de l'arbre amb menor cost, és a dir, en el que la suma de les puntuacions de les comandes no seleccionades és menor.

SMT-Solver

Un *SAT-Solver* és un programa molt eficient el qual permet resoldre problemes de satisfactibilitat booleana. Els *SMT-Solvers* (*Satisfiability Modulo Theories*) *[19] [20]* són una generalització dels *SAT-Solvers* que permeten tractar fórmules més complexes incloent per exemple enters, reals o altres estructures de dades. En el nostre cas, per tal de considerar les restriccions d'estoc utilitzem un SMT-Solver amb aritmètica lineal entera.

Així doncs, per a resoldre el nostre problema amb un *SMT-Solver* el codificaríem com un problema de satisfactibilitat booleana amb aritmètica lineal entera.

D'altra banda, com que el nostre problema és d'optimització i no de satisfactibilitat utilitzem *MAX-SMT* [21], és a dir, un *SMT Solver* que permet trobar una solució òptima. D'aquesta manera podem codificar el nostre problema afegint una funció a minimitzar que serà la suma de les puntuacions de les comandes no seleccionades. Una alternativa seria utilitzar clàusules soft, és a dir, clàusules que tenen un pes associat i que no necessàriament s'han de satisfer. En aquest cas la funció d'optimització del solver intentaria minimitzar la suma dels pesos de les clàusules no satisfetes. Pel nostre problema afegiríem per cada comanda una clàusula soft "es serveix la comanda" amb pes associat "puntuació de la comanda".

9.3.3.4. Tècniques implementades per a resoldre el problema

Les tècniques implementades han estat les dues últimes que hem mencionat, per tant, per una banda hem implementat un algoritme de *Least-Cost Branch&Bound* i per l'altra hem implementat un algorisme que codifica el problema en en clàusules entendibles per *Z3*, l'*SMT Solver* utilitzat.

Hem decidit implementar els dos algoritmes en el llenguatge *Python* per tal de fer una comparativa entre ells. Per a dur a terme aquesta comparativa hem implementat un generador de fitxers de comandes i estocs d'entrada al qual li donem el número

de comandes d'entrada, el total de productes, l'estoc mínim i màxim disponible de cada producte, el número màxim de línies per comanda, la quantitat màxima d'un article en una comanda, les puntuacions mínima i màxima d'una comanda i ens genera aleatòriament dins d'aquests marges el conjunt de comandes i estocs.

En general, per la disponibilitat d'estoc que té l'empresa i el tipus i volum de comandes que aquesta rep en un dia, l'algoritme de *Branch&Bound* es comporta bé i en la majoria dels casos troba la resposta més ràpidament que l'*SMT Solver*, però hi ha altres casos en els que el comportament de l'*SMT Solver* és millor al del Branch&Bound. És per això que cal fer un estudi més profund comparant les dues tècniques.

9.3.3.5. Comparativa de les dos tècniques implementades

Com ja hem comentat anteriorment, els dos algoritmes implementats obtenen una solució òptima i, per tant, el que cal comparar és el temps que tarden en obtenir-la.

Per a dur a terme la comparativa dels temps d'execució entre les dues tècniques implementades s'han fet diverses proves amb valors semblants als de les comandes que solen arribar cada dia a l'empresa a la que va destinada aquest desenvolupament. Aquestes proves s'han executat fixant un temps límit de 30 minuts. Els casos que exhaureixen els 30 minuts sense arribar a la solució no mostren cap valor en els gràfics que mostrem a continuació i, per tant, els veiem reflectits amb una discontinuïtat.

En primer lloc s'han fet les proves amb valors molt ajustats al tipus de comandes que rep l'empresa generant 50 exemples d'aquest tipus:

- 100 comandes
- 50 productes (en total l'empresa té a la venda uns 300 productes però al haver-hi productes molt més populars que d'altres hi ha dies que n'hi poden haver només 50 de diferents entre les comandes)
- Entre 1 i 10 línies per comanda
- Quantitat de 1 a 20 unitats per línia (producte)
- Estoc disponible de cada producte de 0 a 200 unitats
- Puntuació entre 0 i 1 per cada comanda

Els resultats obtinguts han estat els següents:

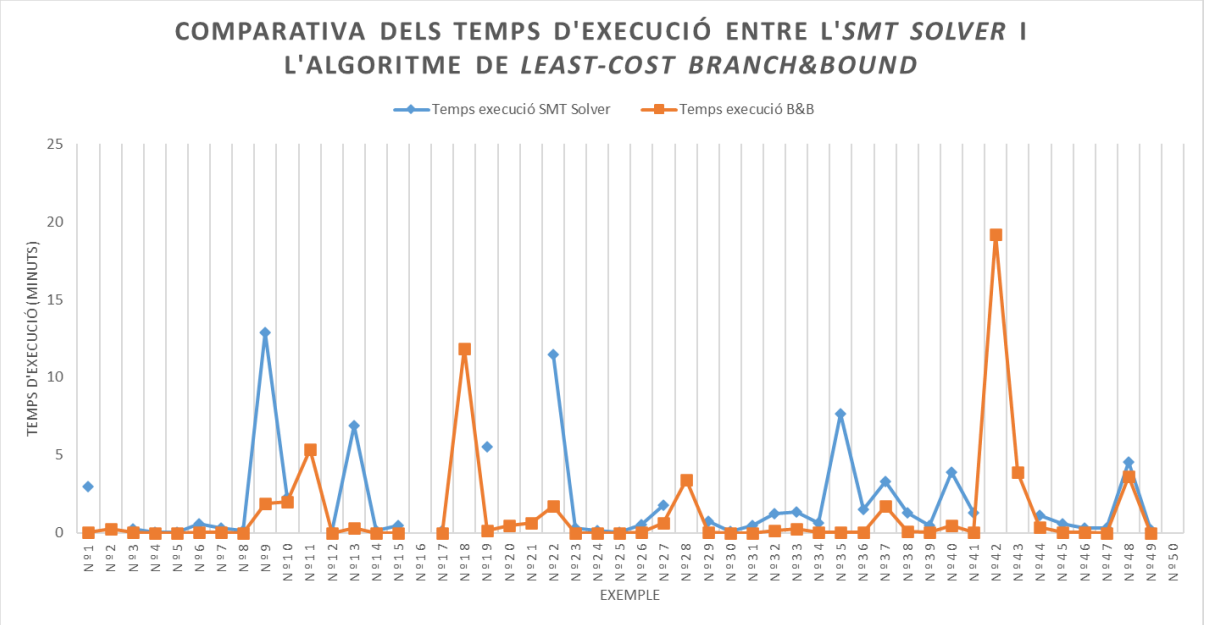


Figura 9.7: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 50 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-200 estoc i 0-1 puntuació. Elaboració pròpia.

A la gràfica anterior podem observar que amb aquest tipus d'exemples el comportament de l'algoritme *Least-Cost Branch&Bound* és clarament millor al de la resolució a través d'un *SMT Solver*.

Seguidament, per a cobrir més casos i acabar de decidir quin algoritme implementem al desenvolupament que ens ha demanat l'empresa, hem estat experimentant amb valors molt semblants als anteriors variant l'estoc disponible dels productes i el nombre de productes diferents que apareixen a les comandes ja que són valors que fàcilment poden variar, ja sigui per les previsions d'estoc de l'empresa o pels diferents productes que apareixen a les comandes.

Variació de l'estoc de cada producte de 0 a 80:

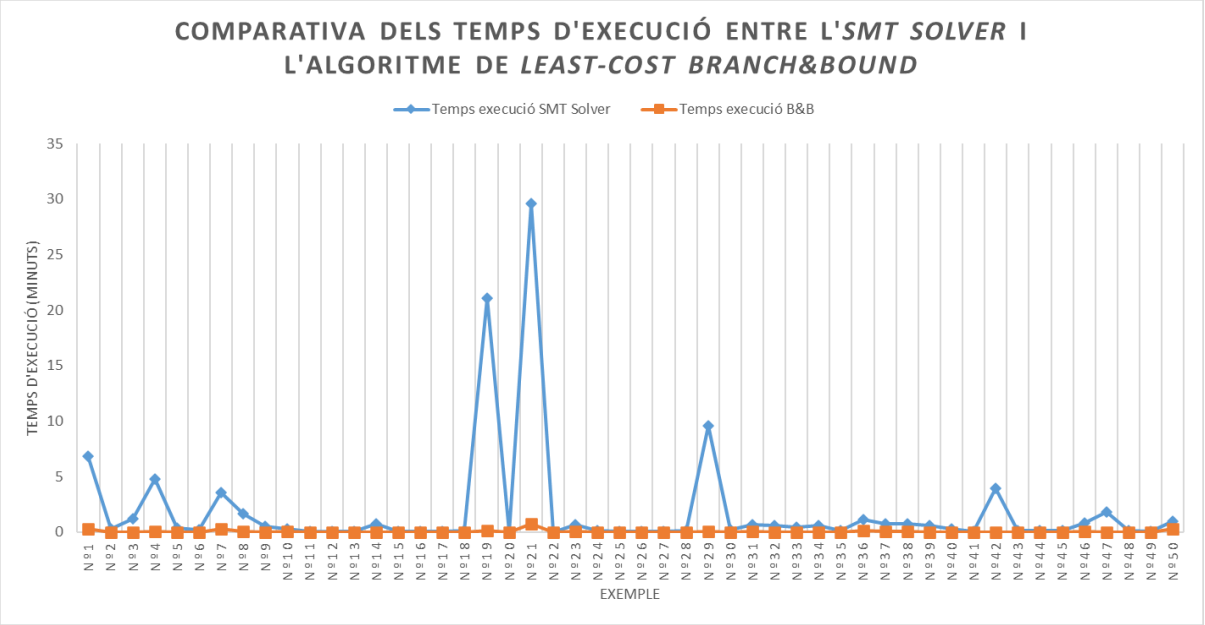


Figura 9.8: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 50 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-80 estoc i 0-1 puntuació. Elaboració pròpia.

Variació de l'estoc de cada producte de 0 a 500:

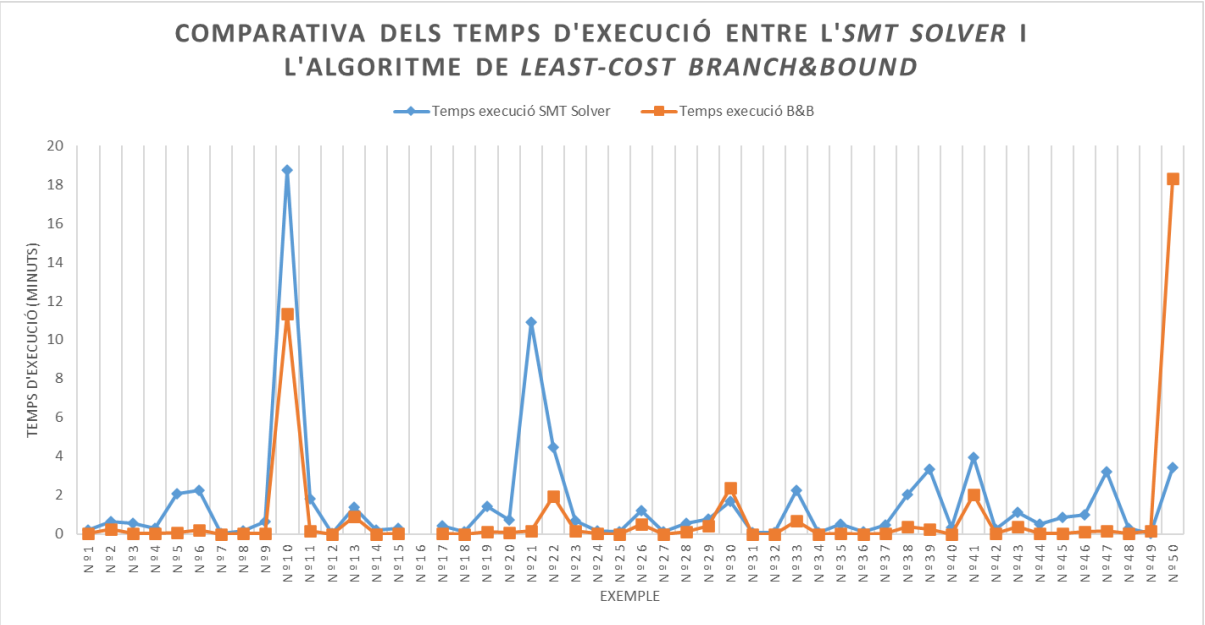


Figura 9.9: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 50 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-500 estoc i 0-1 puntuació. Elaboració pròpia.

Variació del número de productes que apareixen a les comandes de 0 a 25:

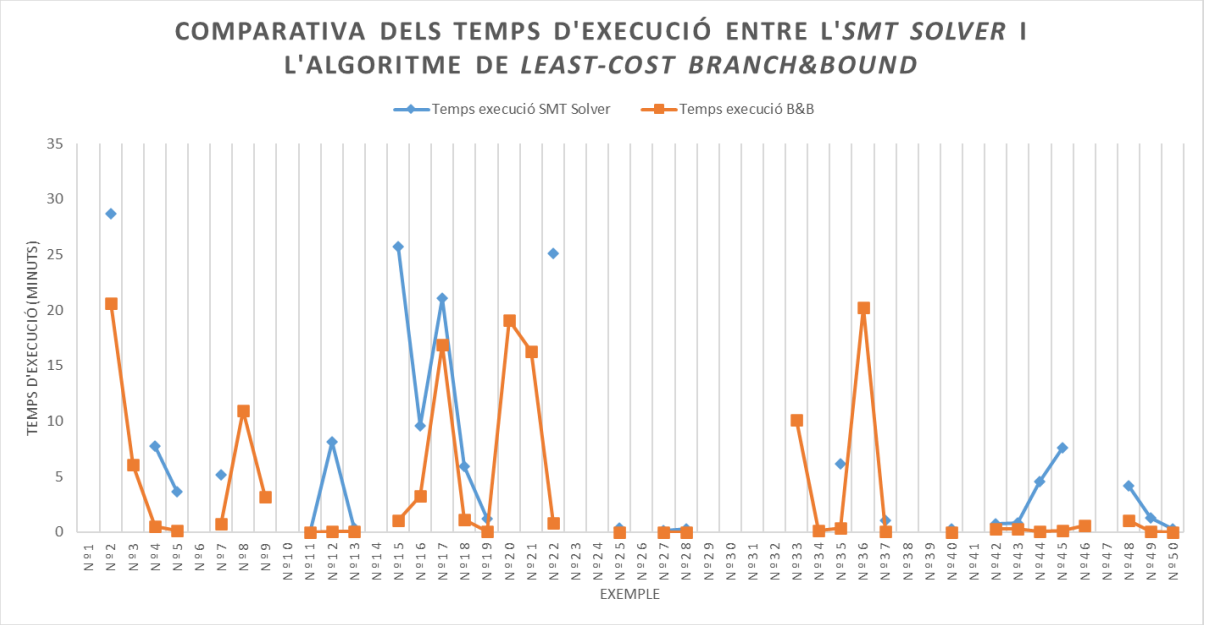


Figura 9.10: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 25 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-200 estoc i 0-1 puntuació. Elaboració pròpia.

Variació del número de productes que apareixen a les comandes de 0 a 75:

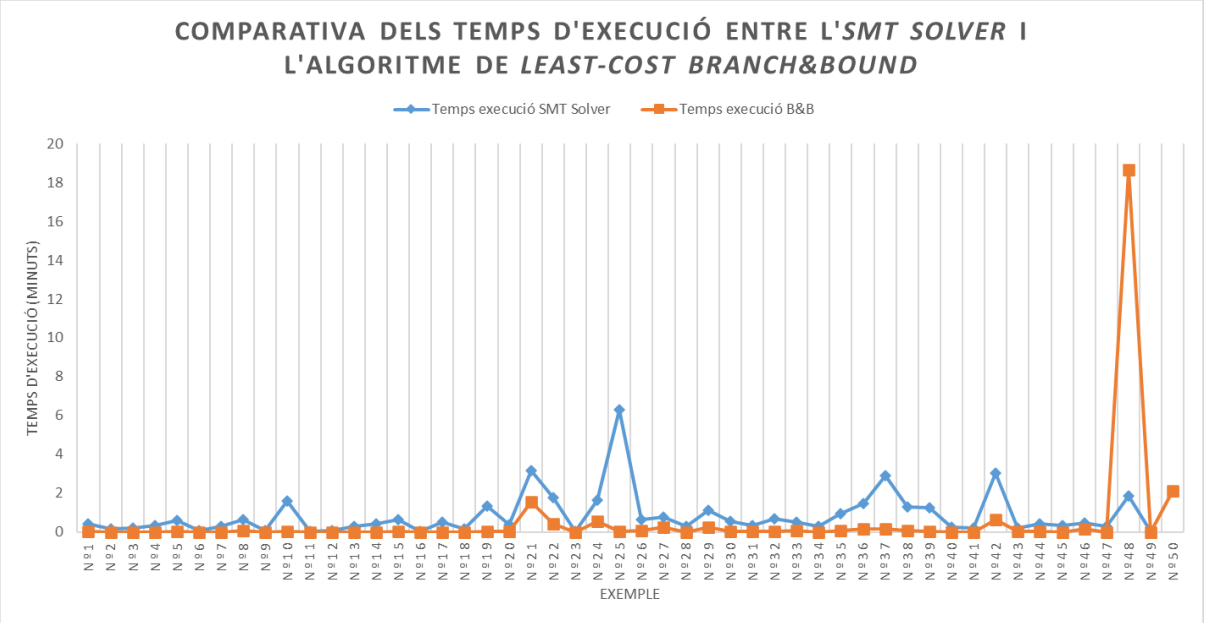


Figura 9.11: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 75 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-200 estoc i 0-1 puntuació. Elaboració pròpia.

Com podem veure, per molt que puguin variar en algun aspecte el tipus de comandes que arriben a l'empresa, en la gran majoria de casos el comportament de l'algoritme *Least-Cost Branch&Bound* segueix sent millor al de la resolució del problema a través del *SMT Solver*, per tant, finalment s'ha decidit que aquest serà

l'algoritme implementat al SAP de l'empresa, de manera que quan executin el procés de priorització des de l'ERP (prement el botó de prioritzar de la pantalla vista a l'apartat *9.2*) aquest serà l'algoritme que assignarà l'estoc de forma intel·ligent.

Tot i això, un cas que no s'ha tingut en compte a l'hora de decidir quin algoritme implementar al desenvolupament per l'empresa on sí és clarament millor resoldre el problema a través de l'*SMT Solver* es dóna si es vol maximitzar el nombre total de comandes servides sense considerar cap priorització, cosa que és el mateix que tenir el mateix pes per a totes les comandes.

La raó per la qual aquest cas no s'ha tingut en compte és perquè no és el que ha demanat l'empresa que precisament té interès en poder prioritzar les comandes en funció del perfil del client. Si es volgués incloure aquest cas, s'hauria plantejat el problema d'una altra manera a l'hora de dissenyar l'heurística utilitzada pel *Branch&Bound*, ja que ara mateix aquesta es basa en les diferències de pesos entre comandes.

Seguidament podem veure els resultats d'executar les proves amb exemples on totes les comandes tenen el mateix pes/puntuació on efectivament, com ja hem comentat, el comportament de la resolució del problema a través de l'*SMT Solver* és clarament millor al de la resolució amb l'algoritme de *Least-Cost Branch&Bound*.

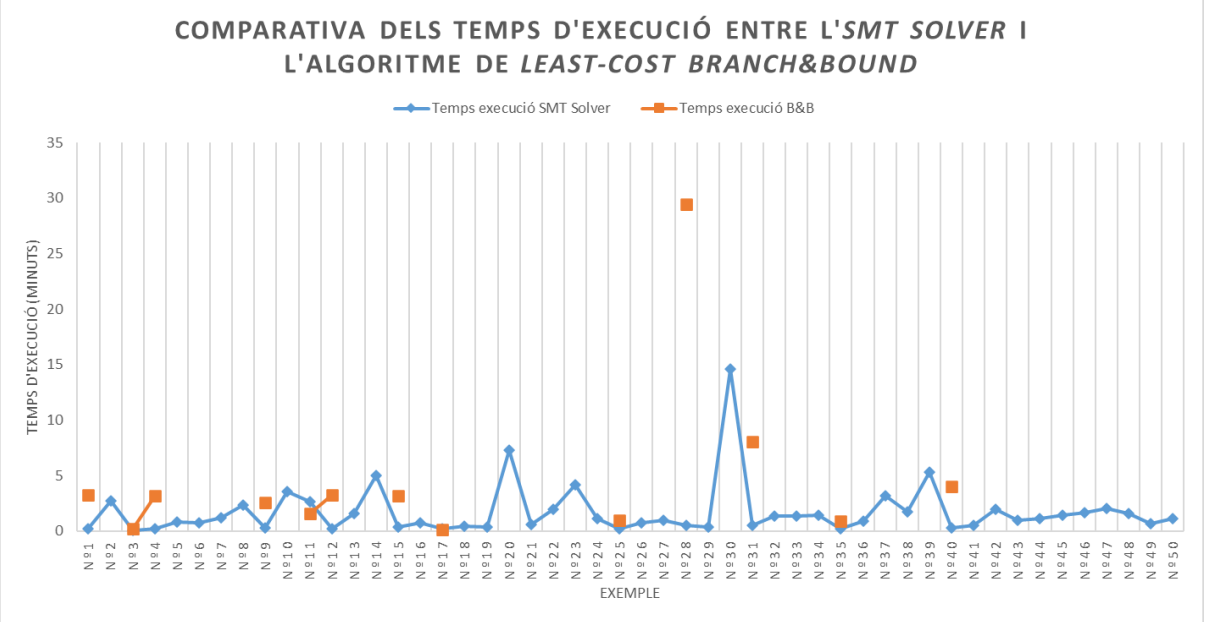


Figura 9.12: Comparativa dels temps d'execució amb 50 exemples de 100 comandes, 50 productes, 1-10 línies/comanda, 1-20 unitats/línia, 0-200 estoc i 1 puntuació. Elaboració pròpia.

9.3.3.6. Codificació del problema en clàusules entendibles per l'*SMT Solver*

En aquest apartat es mostra com s'ha codificat el nostre problema d'optimització en un problema de *Max-SMT*.

Donades *n* comandes ($\mathbf{c_1,...,c_n}$), *k* productes ($\mathbf{p_1,...,p_k}$), les quantitats demanades de cada producte en cada comanda ($\mathbf{q_{11},...,q_{1k},...q_{n1},...,q_{nk}}$) **on** $\mathbf{q_{ij}}$ indica la quantitat demanada de producte j en la comanda i, els estocs d'aquests productes ($\mathbf{ep_1,...,ep_k}$) i la puntuació de cada comanda ($\mathbf{w_1,...,w_n}$), codifiquem el problema de la següent manera:

Literals:

- s_i: es serveix la comanda i (*boolean*)
- sp_{ij}: es serveix el producte j de la comanda i (*boolean*)
- qp_{ij}: quantitat gastada del producte j per la comanda i (*integer*)

Clàusules:

- $\forall i_{1:n} (s_i \rightarrow \forall j_{1:k} sp_{ij})$, si es serveix una comanda llavors es serveixen tots els seus productes.
- $\forall i_{1:n} (\neg s_i \rightarrow \forall j_{1:k} \neg sp_{ij})$, si no es serveix una comanda llavors no es serveix cap producte.
- $\forall i_{1:n} \forall j_{1:k} (sp_{ij} \rightarrow qp_{ij} = q_{ij})$, si es serveix un producte j d'una comanda i, la quantitat gastada d'aquest producte per aquesta comanda és q_{ij} (la quantitat demanada).
- $\forall i_{1:n} \forall j_{1:k} (\neg sp_{ij} \rightarrow qp_{ij} = 0)$, si no es serveix un producte j d'una comanda i, la quantitat gastada d'aquest producte per aquesta comanda és 0.
- $\forall j_{1:k} (\sum_{i=}^{n} qp_{ij} \leq ep_j)$, la suma de les quantitats dels productes de les comandes servides no supera l'estoc d'aquests.

Funció a optimitzar (minimitzar):

- $\sum_{i=1}^{n} (if (s_i, 0, w_i))$, la funció a minimitzar és la suma de les puntuacions de les comandes no servides.

S'ha implementat aquesta codificació utilitzant la API de Z3 per Python. El *SMT Solver* utilitzat és el Z3. Es pot trobar el codi en l'apèndix 2 d'aquesta memòria.

9.3.3.7. Detalls sobre la implementació del *Least-Cost Branch&Bound*

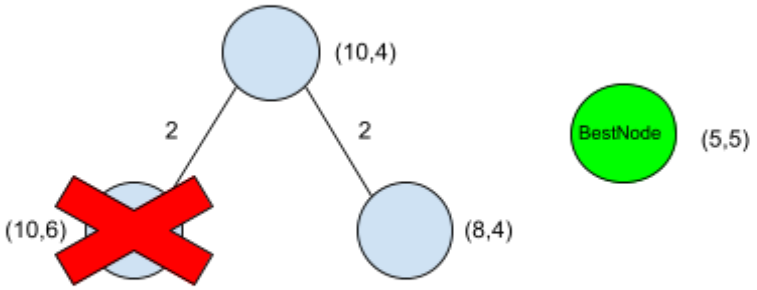
A l'apartat *9.3.3.3 (Anàlisi d'alternatives per a resoldre el problema)* ja hem parlat sobre com funcionen aquest tipus d'algoritmes i seguidament veurem detalls de la nostra implementació.

En el nostre algoritme de *Least-Cost Branch&Bound* l'ordre de visita dels nodes el marca una cua de prioritat que conté els nodes oberts ordenats de menor a major cost màxim. Com ja hem mencionat abans, el primer node que es visita és el *dummy*. Per cada node que visitem es poden afegir dos fills, el de la dreta es correspon al node resultant de seleccionar la comanda amb major puntuació (la primera de la llista de comandes pendents de seleccionar a partir d'aquest node que es manté ordenada de major a menor puntuació) mentre que el de l'esquerra es correspon a no seleccionar-la.

Cada un dels dos nodes afegits es poda (és a dir, no s'afegeix a la cua de nodes oberts) si el cost mínim que pot tenir una solució a partir d'aquest node és superior al cost de la millor solució obtinguda fins al moment, seguidament en podem veure un exemple:

En tots els exemples que veurem d'aquí en endavant, al costat de cada node s'indica el cost màxim i el cost mínim El número que es troba al costat de cada línia que uneix el node pare amb els nodes fills es correspon a la puntuació de la comanda a tractar (la qual en el cas del fill esquerre no es selecciona i en el cas del fill dret sí).

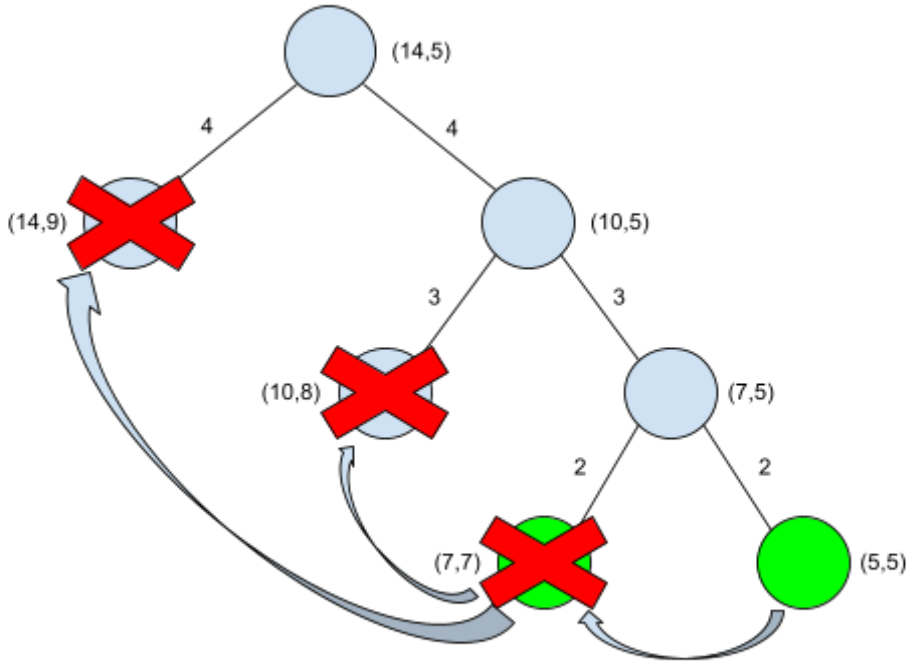
En aquest exemple ens trobem amb la següent situació en un pas de l'algoritme on es visita un node i s'han de generar els seus fills:



En aquesta situació, no seleccionar la comanda a tractar (generar fill esquerre) amb una puntuació igual a 2 implica que aquest node tindrà un cost mínim superior al cost de la millor solució fins el moment que es troba al *BestNode*, per tant el podem.

En cas contrari, per cada node afegit si aquest és una fulla passa a ser la millor solució i es poden aquells nodes oberts els quals el cost mínim que es pot obtenir a partir d'aquests és superior o igual al cost d'aquesta solució; en canvi, si aquest no és una fulla simplement s'afegeix a la cua de prioritat de nodes oberts ordenada de menor a major cost màxim. Només ens cal fer aquesta comprovació de poda quan arribem a una solució ja que sempre visitem els nodes oberts de menor a major cost màxim i quan generem els fills d'un node aquests sempre tindran un cost màxim menor o igual al del pare (node que acabem de tractar, per tant node amb menor cost màxim fins el moment). Exemple:

Ens trobem a un pas de l'algoritme en la següent situació quan afegim els dos nodes fulla marcats de color verd.



En aquesta situació es poden tots els nodes oberts tatxats amb una creu a causa de trobar una solució (node fulla) amb un cost inferior o igual al cost mínim d'aquests. Cal notar que els dos nodes superiors es poden podar abans però no fa falta fer-ho fins arribar a una solució.

A continuació podem veure una captura del codi en *VisualBasic* on, per una banda, la funció *possibleSolution* s'encarrega de podar el node generat quan a partir d'ell no es pot obtenir una solució millor que l'actual (com hem vist al primer exemple), i per altra banda, la funció *pruneOpenNodes* és l'encarregada de podar aquells nodes oberts amb un cost mínim superior o igual al cost de la millor solució (vist al segon exemple).

```vb
Private Function GetOrdersToServe(ByVal listOrdersSorted As List(Of Tuple(Of Double, Integer))) As Node
    Dim maxCost As Double = 0
    Dim firstNode As Node
    Dim currentNode As Node
    Dim nodeLeft As Node
    Dim nodeRight As Node
    Dim leftInserted As Boolean
    Dim lOpenNodes As List(Of Node) = New List(Of Node)
    Dim bestCurrentSolution As Node = Nothing
    '
    For Each com As Tuple(Of Double, Integer) In listOrdersSorted
        maxCost += com.Item1 'costMax: cost total si no servim cap comanda
    Next
    '
    Node.dOrders = Me.dOrders
    Node.dLinOrd = Me.dLinOrders
    firstNode = New Node(New List(Of Integer), maxCost, maxCost, Me.dStocks, listOrdersSorted, False, False) 'Node dummy
    lOpenNodes.Add(firstNode)
    '
    While lOpenNodes.Count > 0
        currentNode = lOpenNodes(0)
        lOpenNodes.RemoveAt(0)
        nodeLeft = currentNode.genLeftNode()
        leftInserted = False
        If nodeLeft.possibleSolution(bestCurrentSolution) Then
            If nodeLeft.IsLeaf() Then
                bestCurrentSolution = nodeLeft 'Si una possible solució és una fulla, aquesta passa a ser la millor solució
                lOpenNodes = pruneOpenNodes(lOpenNodes, nodeLeft.getMaxCost())
            Else
                lOpenNodes.Insert(0, nodeLeft)
                leftInserted = True
            End If
        End If
        '
        nodeRight = currentNode.genRightNode()
        If nodeRight.possibleSolution(bestCurrentSolution) Then
            If nodeRight.IsLeaf() Then
                bestCurrentSolution = nodeRight 'Si una possible solució és una fulla, aquesta passa a ser la millor solució
                lOpenNodes = pruneOpenNodes(lOpenNodes, nodeRight.getMaxCost())
            Else
                If leftInserted And nodeRight.getMaxCost() > nodeLeft.getMaxCost() Then
                    lOpenNodes.Insert(1, nodeRight)
                Else
                    lOpenNodes.Insert(0, nodeRight)
                End If
            End If
        End If
    End While
    '
    Return bestCurrentSolution
End Function
```

Figura 9.13: Captura de la funció de creació d'un node en VB.NET. Elaboració pròpia.

A continuació podem veure com es fa el desplegament de cada node que tractem. Per a cada node, a més de tenir el cost màxim i mínim com hem comentat anteriorment, també hi guardem la següent informació:

- Llista de comandes seleccionades fins al moment.
- Llista de comandes pendents de considerar ordenades de major a menor puntuació (que correspon al cost de no servir-la). La comanda que es tracta al desplegar el node és la primera de la llista, és a dir, la de major puntuació.
- Informació dels estocs actualitzada d'acord amb les comandes ja seleccionades prèviament en la branca de l'arbre que ens ha portat al node actual.

Quan parlem de desplegar un node ens referim a generar el fill de l'esquerra el qual es correspon a no seleccionar la comanda considerada, i el fill de la dreta que es correspon a seleccionar-la.

Guardar la informació dels estocs a cada node ens permetrà, en alguns casos, prendre decisions per avançat respecte la resta de comandes de la llista de comandes pendents reduint així la profunditat de l'arbre de cerca que ens queda a sota del node generat i, també, descartar (si és el cas) nodes que no poden superar la millor solució actual amb més antelació.

Seguidament exposem com es duu a terme la generació dels dos nodes fills a partir del node que s'està tractant.

El node de l'esquerra es correspon al cas de no seleccionar la comanda (la primera de la llista de comandes pendents a considerar del node tractat ordenada de major a menor puntuació), per tant, el cost mínim d'aquest és igual al del pare sumant-hi la puntuació d'aquesta comanda i el màxim es queda igual. Seguidament, donat que s'ha descartat una comanda pot passar que de la resta de comandes pendents de considerar n'hi hagi algunes que ja es puguin servir. Si és el cas, aquestes s'afegeixen a la llista de comandes seleccionades, s'eliminen de la llista de comandes pendents de tractar reduint així la profunditat de l'arbre de cerca i s'actualitza el cost màxim del node generat restant el cost d'aquestes comandes.

El node de la dreta es correspon al cas de seleccionar la comanda, per tant, el cost mínim no es modifica (respecte el del pare), el cost màxim es decrementa amb el cost d'aquesta comanda i s'actualitzen els estocs disponibles. Seguidament, donat que s'ha seleccionat una comanda, pot passar que de la resta de comandes pendents de considerar n'hi hagi algunes que no es puguin servir a causa de l'estoc restant. En aquest cas, aquestes es treuen de la llista de comandes pendents de considerar i s'afegeix el seu cost al cost mínim del node. Seguidament, el que pot passar si es descarta alguna comanda és que de les comandes restants n'hi hagi altres que segur que es poden servir i, en aquest cas es procedeix igual com s'ha fet amb el node de l'esquerra, s'afegeixen a la llista de comandes seleccionades i s'actualitza el cost màxim.

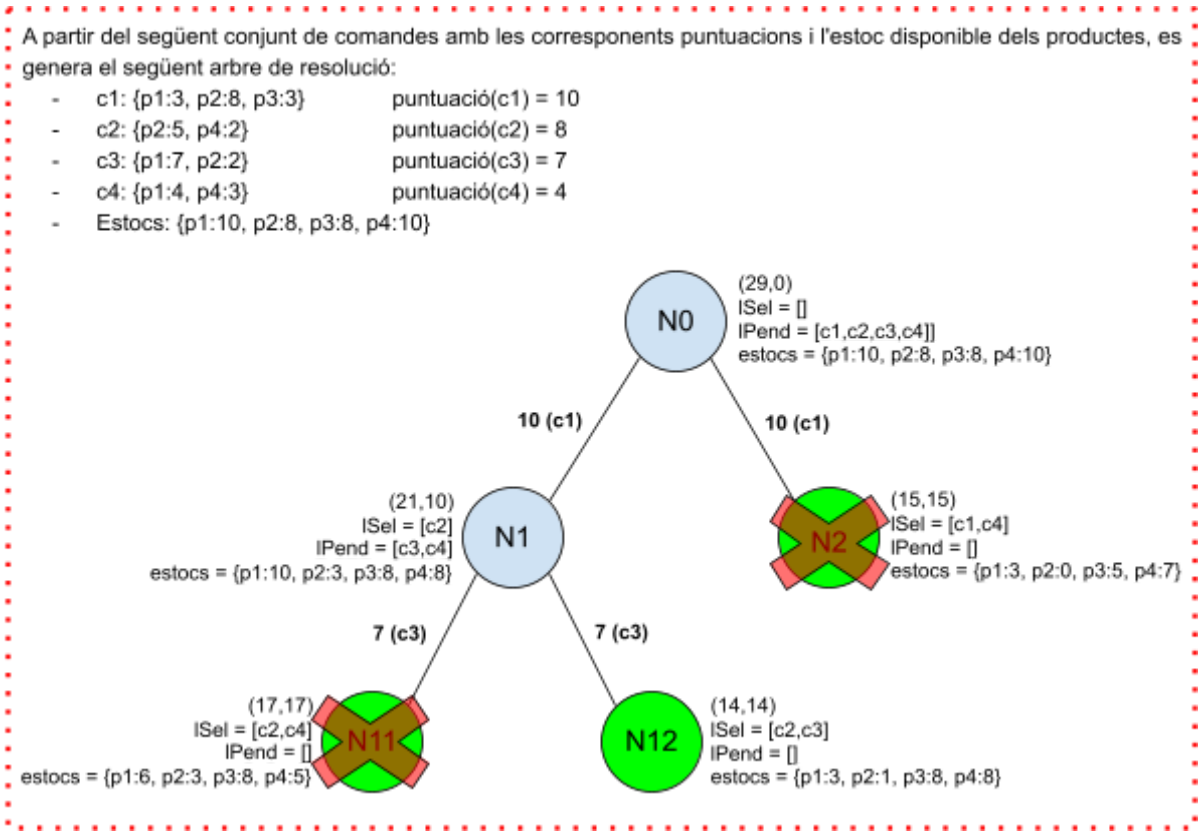
A continuació podem veure una captura del codi en *VisualBasic* on s'aprecia el procés que segueix la creació de cada node. Els nodes creats per l'esquerra (no es serveix la comanda a tractar) es criden amb *checkAheadDone=True* i *checkAheadNo=False* i viceversa pels nodes creats per la dreta.

```
Public Sub New(l, mC, bRC, dStocks, lToExp, checkAheadDone, checkAheadNo)
    Me.selected = l
    Me.maxCost = mC
    Me.bestRestCost = bRC
    Me.dStocks = dStocks
    Me.lToExplore = lToExp
    '
    If Me.lToExplore.Count > 0 Then
        If checkAheadNo Then 'Cal comprovar si hi ha comandes que no es poden servir.
            If Me.UpdateNoServe() Then
                Me.UpdateDone() 'Si deixem de servir alguna comanda cal comprovar si a causa d'això alguna es pot servir.
            End If
        Else
            If checkAheadDone Then 'Cal comprovar si hi ha comandes que segur que es poden servir.
                Me.UpdateDone()
            End If
        End If
    End If
    '
    Me.maxCost = Math.Round(Me.maxCost, 4)
    Me.bestRestCost = Math.Round(Me.bestRestCost, 4) 'Necessari, sinó apareixen decimals
End Sub
```

Figura 9.14: Captura de la funció de creació d'un node en VB.NET. Elaboració pròpia.

Per tant, amb el tractament que realitzem al generar els nodes fills, d'acord amb les restriccions d'estoc que es guarden actualitzades a cada node, es poden prendre decisions sobre les comandes pendents de tractar disminuint així la profunditat de l'arbre de cerca a partir dels nodes generats.

Seguidament mostrem aquest funcionament a partir d'un exemple:



Explicació de l'exemple:

- Quan es genera el node N1 s'analitzen les comandes c2, c3 i c4: la comanda c2 segur que ja es pot servir donat que inclou els productes p2 i p4 i la suma de quantitats demanades d'aquests dos productes a les comandes c2, c3 i c4 és inferior a l'estoc restant. Per tant, s'afegeix c2 a la llista de comandes seleccionades, es treu de la llista de les pendents i s'actualitzen el cost màxim i els estocs.
- Quan es genera el node N2 s'analitzen les comandes c2, c3 i c4: les comandes c2 i c3 no es podran servir donat que no hi ha suficient estoc del producte p2. Només queda la comanda p4 i es pot servir donat que hi ha suficient estoc. Per tant, s'afegeix c4 a la llista de comandes seleccionades, es treuen c2, c3 i c4 de la llista de comandes pendents i s'actualitzen el cost màxim, mínim i els estocs.
- Com que N2 es queda sense cap comanda pendent de tractar podem dir que N2 és una possible millor solució la qual inclou les comandes [c1,c4] amb cost 15. Seguidament, al haver arribat a una possible solució, es comprova si es pot podar algun node de la llista de nodes oberts on només hi ha N1, el

qual no es poda ja que el cost mínim d'aquest (10) és inferior al de la millor solució actual (15).

- Al tractar el node N1, procedint com s'ha vist anteriorment, es generen el fill de l'esquerra (N11) i el de la dreta (N12).
- El node N11 es poda ja que el seu cost és 17, per tant, superior al de la millor solució actual (15).
- El node N12 passa a ser la millor solució la qual inclou les comandes [c2,c3] amb cost 14 descartant N2, la que fins ara era la millor solució.

A l'apèndix 1 es mostra el comportament de l'algoritme amb un exemple complert.

Amb aquesta implementació s'ha aconseguit un algoritme que, en general, es comporta molt bé amb el tipus de comandes i estocs de l'empresa a qui va destinat. En els casos en que l'algoritme requereixi més temps del que s'hagi establert com a temps màxim per a l'execució, es pot donar la millor solució obtinguda fins al moment aconseguint així una bona aproximació generalment millorant la que s'obtindria amb un algoritme voraç.

10. Conclusions

En aquest projecte s'ha dut a terme un desenvolupament que, per una banda permet introduir a l'ERP SAP de l'empresa al que va dirigit les comandes provinents d'un servei web i, per altra banda, els permet prioritzar l'assignació de l'estoc disponible sobre el conjunt de comandes d'entrada en funció del perfil dels clients.

El disseny i la implementació d'aquest sistema permeten a l'empresa configurar els paràmetres de priorització com més els convé de forma senzilla i clara i seguidament executar el procés de priorització de l'estoc. Un cop executat aquest procés s'assignen magatzems als productes de les comandes seleccionades tenint en compte la disponibilitat que aquests tenen de cada producte.

Així doncs, aquest desenvolupament permetrà a l'empresa d'ara en endavant prioritzar l'assignació de l'estoc a les comandes que rep el servei web de forma ràpida, senzilla i eficaç i, per tant, podem dir que s'han assolit els objectius del projecte.

Durant la realització d'aquest projecte s'han adquirit coneixements en diversos àmbits, especialment en l'àmbit del desenvolupament i la gestió de projectes, en el de tècniques d'optimització i en el del desenvolupament de funcionalitats a l'ERP SAP Business One.

10.1. Millores i treball futur

En el sistema desenvolupat, l'assignació de l'estoc dels magatzems a les comandes seleccionades s'ha realitzat tal i com ho fan actualment a l'empresa client. Aquesta assignació es realitza de forma seqüencial servint tot el que es pot d'un primer magatzem i, a mesura que s'acaba l'estoc, s'agafa del següent magatzem. D'aquesta manera, la preparació de cada producte d'una comanda pot involucrar fins als quatre magatzems de què disposen.

Es vol proposar a l'empresa completar el projecte amb una millora d'aquesta assignació intentant minimitzar el nombre de magatzems involucrats en la preparació de cada comanda. S'ha realitzat la implementació en l'entorn de proves desenvolupat i s'està pendent de realitzar proves per analitzar el temps que requereix la solució implementada en el tipus de comandes de l'empresa abans de fer-los la proposta.

La solució implementada consisteix en codificar el problema com un problema de *Max-SMT*. El problema que volem resoldre és el següent: donat un conjunt de comandes, que sabem que es poden servir amb l'estoc disponible en el conjunt dels magatzems, assignar la preparació dels productes de cada comanda als diversos magatzems, intentant minimitzar, en conjunt, el nombre de magatzems involucrats en la preparació de cadascuna d'elles.

Seguidament mostrem quin és el problema i la codificació a *Max-SMT* realitzada per resoldre'l.

Donades n comandes $\{c_1,...,c_n\}$, s magatzems $\{m_1,...,m_s\}$, m productes $\{p_1,...,p_m\}$, sigui lin_{ij} la quantitat demanada del producte j a la comanda i per tot i Є [1..n], j Є [1..m], i sigui $stock_{kj}$ la quantitat d'estoc del producte j en el magatzem k per tot k Є [1..s], j Є [1..m] codifiquem el problema de la següent manera:

Literals:

- qp_{ijk}: quantitat del producte j de la comanda i servida del magatzem k (*integer*).
- $served_{ik}$: part d'algun producte de la comanda i es serveix des del magatzem k (*boolean*).

Clàusules:

- $\forall i_{1:n} \forall j_{1:m} \forall k_{1:s} (0 \leq qp_{ijk} \leq lin_{ij})$, la quantitat servida d'un producte a una comanda en cada magatzem és un valor entre 0 i la quantitat demanada.
- $\forall i_{1:n} \forall j_{1:m} (\sum_{k=1}^{s} qp_{ijk} = lin_{ij})$, la quantitat servida d'un producte a una comanda entre tots els magatzems coincideix amb la quantitat demanada.
- $\forall j_{1:m} \forall k_{1:s} (\sum_{i=1}^{n} qp_{ijk} \leq stock_{kj})$, la quantitat servida d'un producte des d'un magatzem considerant totes les comandes mai supera l'estoc del producte en aquest magatzem.
- $\forall i_{1:n} \forall k_{1:s} (served_{ik} \; iff \; (\exists j_{1:m} \; qp_{ijk} > 0)$, una comanda és servida des d'un magatzem si i només si es serveix part d'algun dels seus productes demanats a la comanda.

Funció a optimitzar (minimitzar):

- $\sum_{i=1}^{n} \sum_{k=1}^{s} (if \; (served_{ik}, \; 1, \; 0))$, la funció a minimitzar és la suma del nombre de magatzems utilitzats per a servir cada comanda.

Referències

[1] *Seidor Business One: Software ERP SAP:* https://www.seidor.com/es-es/servicios-para-pymes [en línia] (consulta 25/02/2022)

[2] *SAP Business One | ERP Software for Small Businesses:* https://www.sap.com/spain/products/business-one.html [en línia] (consulta 25/02/2022)

[3] Reducción de problemas de generación de horarios a SAT: https://upcommons.upc.edu/bitstream/handle/2117/87538/R08-19.pdf (consulta 03/03/2022)

[4] SAP Business One | ERP Software for Small Businesses: https://www.sap.com/products/business-one.html [en línia] (consulta 06/03/2022)

[5] Descargas de SQL Server | Microsoft: https://www.microsoft.com/es-es/sql-server/sql-server-downloads [en línia] (consulta 06/03/2022)

[6] Google Drive: https://drive.google.com/drive/u/0/my-drive [en línia] (consulta 06/03/2022)

[7] Canva.com - Canva: https://www.canva.com [en línia] (consulta 06/03/2022)

[8] Visual Studio: IDE and Code Editor for Software Developers: https://visualstudio.microsoft.com/es/ [en línia] (consulta 06/03/2022)

[9] Internet Information Services (IIS) 10.0 Express - Microsoft: https://www.microsoft.com/es-es/download/details.aspx?id=48264 [en línia] (consulta 05/04/2022)

[10] Z3 - Microsoft Research https://www.microsoft.com/en-us/research/project/z3-3/ [en línia] (consulta 14/04/2022)

[11] Calculadora Salarial | Trabajadores: Guía del Mercado Laboral 2022 - Hays: https://guiasalarial.hays.es/trabajador/calculadora-salarial [en línia] (consulta 17/03/2022)

[12] Silla Ergonómica MARKO, soporte lumbar, en negro - Ofisillas.es: https://www.ofisillas.es/silla-ergonomica-marko-soporte-lumbar-mecanismo-sincronizado-base-aluminio-en-negro.html [en línia] (consulta 11/03/2022)

[13] MICKE Escritorio, negro-marrón 105x50 cm - IKEA: https://www.ikea.com/es/es/p/micke-escritorio-negro-marron-10244743/ [en línia] (consulta 11/03/2022)

[14] El preu de la llum, actualitzat: l'hora més cara i la més barata d'avui (rac1.cat): https://www.rac1.cat/economia/20210831/4101395871834/preu-llum-factura-record-hora-mes-cara-barata-electricitat.html [en línia] (consulta 11/03/2022)

[15] Qüestionari Genèric Estudiants d'Enginyeria: https://docs.google.com/forms/d/e/1FAIpQLSfVgBxcxZfh7pB_OVRUNGQmRpFDFlhAskukNcpQBowLRF4-sA/viewform [en línia] (consulta 12/03/2022)

[16] Algoritmos voraces en JavaScript: Mochila fraccionable (wextensible.com): https://www.wextensible.com/temas/voraces [en línia] (consulta 07/04/2022)

[17] Greedy Algorithms (General Structure and Applications) - GeeksforGeeks: https://www.geeksforgeeks.org/greedy-algorithms-general-structure-and-applications/ [en línia] (consulta 07/04/2022)

[18] DAA- Least cost branch and bound | i2tutorials: https://www.i2tutorials.com/design-and-analysis-of-algorithmsdaa-tutorial/daa-least-cost-branch-and-bound/ [en línia] (consulta 14/04/2022)

[19] Reformulation of constraint models into SMT (tesisenred.net): https://www.tesisenred.net/bitstream/handle/10803/392163/tmps1de1.pdf?sequence=8 [en línia] (consulta 02/05/2022)

[20] barret2_smt.pdf (mpg.de): https://resources.mpi-inf.mpg.de/departments/rg1/conferences/vtsa08/slides/barret2_smt.pdf [en línia] (consulta 02/05/2022)

[21] Program Analysis using SMT and MAX-SMT (upc.edu): https://www.cs.upc.edu/~albert/papers/LOPSTR-Slides.pdf [en línia] (consulta 02/05/2022)

[22] Llei Orgànica de protecció de dades de caràcter personal - Viquipèdia, l'enciclopèdia lliure (wikipedia.org): https://ca.wikipedia.org/wiki/Llei_Orgànica_de_protecció_de_dades_de_caràcter_personal [en línia] (consulta 15/05/2022)

Apèndix 1: Exemple de la priorització de l'estoc

Exemple resolt amb l'algoritme de *Branch&Bound* implementat amb 25 comandes, 20 productes, 1-3 línies/comanda, 1-20 unitats/línia, 10-100 estoc.

A la següent taula es mostren els estocs dels 20 productes en els 3 magatzems.

	Items																			
Magatzems	I001	I002	I003	I004	I005	I006	I007	I008	I009	I010	I011	I012	I013	I014	I015	I016	I017	I018	I019	I020
m1	6	66	9	4	0	7	3	6	5	14	41	2	14	32	24	1	33	1	12	2
m2	13	2	40	33	69	8	18	0	11	4	0	11	0	39	25	58	2	6	18	45
m3	45	6	4	33	0	2	3	19	31	40	0	56	8	6	7	3	10	5	10	3
Total	64	74	53	70	69	17	24	25	47	58	41	69	22	77	56	62	45	12	40	50

Seguidament, es mostren les comandes amb els seus productes i la seva puntuació. S'ha indicat en colors el resultat del preprocés: en vermell la comanda descartada per manca d'estoc, i en verd les comandes que s'inclouen directament a la solució (donat que els productes que contenen tenen suficient estoc per servir totes les comandes). Els productes que tenen suficient estoc per servir totes les comandes apareixen marcats en gris.

		Items																		
Comandes	**Puntuació**	I001	I002	I003	I004	I005	I006	I007	I008	I009	I010	I011	I012	I013	I014	I015	I016	I017	I018	I019	I020
333	2,3567								5		1								2		
334	1,1854	8															10				
335	2,0581						14														
336	2,9266			13							16					12					
337	3,008					20			5									16			
338	0,8276						10													6	
339	0,3691					7														9	6
340	3,9752								14	11											
341	1,5176			8							12				18						
342	1,0922										11								15		
343	3,8478						4														
344	2,2412		11												4	5					
345	3,3255						11												7		
346	0,2722		8					7						6							
347	1,3854	13																			
348	3,2451								14						19		5				
349	0,4652					5													10		
350	3,4157				19											1				16	
351	0,0559											6									
352	3,2252								1								11				
353	3,6508				18									17							
354	2,7239				11																
355	2,8394				6			19						10							
356	3,2718		5												19						
357	2,8925				16								10						5		
358	2,0583							6													

En acabar el preprocés s'han generat dues particions que s'han resolt de forma independent. Cal notar que els productes amb suficient estoc no es consideren a l'hora de fer les particions. Les comandes seleccionades de cada partició es mostren en verd i les descartades en vermell.

Comandes	Puntuació	I001	I002	I003	I004	I005	I006	I007	I008	I009	I010	I011	I012	I013	I014	I015	I016	I017	I018	I019	I020
333	2,3567								5		1								2		
335	2,0581						14														
337	3,008					20			5									16			
338	0,8276						10													6	
340	3,9752								14	11											
343	3,8478						4														
345	3,3255						11												7		
348	3,2451								14						19		5				
349	0,4652					5													10		
352	3,2252								1								11				
357	2,8925				16								10						5		
346	0,2722		8					7						6							
353	3,6508				18									17							
355	2,8394				6			19						10							
358	2,0583							6													

La suma de la puntuació de les comandes seleccionades és 45,0759 i el cost de la solució obtinguda és 12,0643 (és a dir, la suma de les puntuacions de les comandes no seleccionades). Per últim, es mostra la solució final amb la distribució dels productes de les comandes entre els 3 magatzems.

Comanda	Producte	Magatzem	Quantitat
334	I001	2	8
	I016	2	10
336	I003	2	13
	I010	3	16
	I015	2	12
337	I005	2	20
	I008	1	5
	I017	1	16
339	I005	2	7
	I019	2	9
	I020	2	6
340	I008	3	14
	I009	3	11
341	I003	1	8
	I010	1	12
	I014	1	18
343	I006	1	4
344	I002	1	11
	I014	1	4
	I015	1	5
345	I006	2	8
	I006	1	3
	I018	2	6
	I018	1	1
347	I001	3	13
350	I004	2	19
	I015	3	1
	I019	2	6
	I019	3	10
351	I011	1	6
352	I008	1	1
	I016	2	11
353	I004	1	3
	I004	3	15
	I013	1	9
	I013	3	8
354	I004	2	11
356	I002	1	5
	I014	2	19
357	I004	3	16
	I012	3	10
	I018	3	5
358	I007	2	6

Apèndix 2: Codificació *a SMT* i resolució mitjançant Z3

```
from z3 import *
from tools import *
import sys

def qty(d,p):
	if p in d:
		return d[p]
	else:
		return 0

def getSum(d,p):
	return sum(list(map(lambda x:qty(x[1],p), list(d.items()))))

def solvePartition(lOrders,lStocks,lProducts):
	served=[]
	for i in range(0,len(lOrders)):
		served.append(Bool(f's_{i}'))

	servedProducts=[]
	for i in range(0,len(lOrders)):
		l=[]
		for j in range(0,len(lProducts)):
			l.append(Bool(f'sp_{i}_{j}'))
		servedProducts.append(l)

	qtyProdCom=[]
	for i in range(0,len(lOrders)):
		l=[]
		for j in range(0,len(lProducts)):
			l.append(Int(f'qp_{i}_{j}'))
		qtyProdCom.append(l)

	o = Optimize()

	#si es serveix una comanda, es serveixen tots els seus productes
	for i in range(0,len(lOrders)):
		o.add(Implies(served[i], And([servedProducts[i][j] for j in
	range(0,len(lProducts))])))

	#si no es serveix una comanda, no es serveix cap producte
	for i in range(0,len(lOrders)):
		o.add(Implies(Not(served[i]), And([Not(servedProducts[i][j]) for j in
	range(0,len(lProducts))])))

	#si un producte d'una comanda es serveix, la quantitat d'estoc gastada d'aquest
	producte per aquesta comanda és la demanada.
	#Si no es serveix, la quantitat és 0.
	for i in range(0,len(lOrders)):
		for j in range(0,len(lProducts)):
```

```
                        if lProducts[j] in dLinOrders[lOrders[i]]:
                                q=dLinOrders[lOrders[i]][lProducts[j]]
                        else:
                                q=0
                o.add(Implies(servedProducts[i][j], qtyProdCom[i][j]==q ))
                o.add(Implies(Not(servedProducts[i][j]), qtyProdCom[i][j]==0 ))

        #La quantitat utilitzada d'un producte no pot ser superior al seu estoc
        for j in range(0,len(lProducts)):
                o.add(sum([qtyProdCom[i][j] for i in range(0,len(lOrders))])<=lStocks[j])

        #funció a minimitzar
        pCost = o.minimize(Sum([If(served[i], 0, dWeightOrders[lOrders[i]])  for i in
        range(0,len(lOrders))]))

        solPartition=[]
        if o.check() == sat:
                m = o.model()
                for i in range(0,len(lOrders)):
                        if m[served[i]]:
                                solPartition.append(lOrders[i])
        return (solPartition, pCost.value().as_string())

def convert(valStr):
        l=valStr.strip().split("/")
        if len(l)==1:
                return float(l[0])
        else:
                return float(l[0])/float(l[1])

if __name__=="__main__":
        #S'obtenen les dades a partir dels fitxers generats.
        #S'indica el directori i el número de l'exemple a tractar
        (dWeightOrders0,dLinOrders0,dStocks0)=getData(sys.argv[1],sys.argv[2])
        #La funció simplify és comú als dos mètodes (es troba a tools).
        #Elimina les comandes que no es poden servir
        (lTrivSol,dWeightOrders,dLinOrders,dStocks)=simplify(dWeightOrders0,dLinOrders0,
        dStocks0)
        lOrders=list(dLinOrders.keys())
        lStocks=list(dStocks.values())
        lProducts=list(dStocks.keys())
        solution=lTrivSol
        costSolution=0
        if len(lOrders)>0:
                #La funció partition és comú als dos mètodes (es troba a tools).
                #Genera les particions
                lpartitions=partition(dWeightOrders,dLinOrders)
                for part in lpartitions:
                        print("solving",part)
                        (solutionPartition, costPart) = solvePartition(part,lStocks,lProducts)
                        costPart=convert(costPart)
                        print("solution partition:",solutionPartition)
                        print("cost partition:",costPart)
                        solution+=solutionPartition
                        costSolution+=costPart
```

```
solution.sort()
#comprovació pel període de proves
(correct,dEstocsRest,score)=checkSolution(solution,dWeightOrders0,dLinOrders0,dS
tocks0)
if correct:
        print("\nSOLUTION: ",solution)
        print("Cost:",costSolution)
        print("Score: ",score)
else: #en totes les proves realitzades mai s'ha trobat una solució incorrecta
        print("INCORRECT SOLUTION")
```