MASTER'S THESIS 2024

Artikelplaceringssystem för industriell robot med constraintprogrammering

Astrid Dymling, Martin Nybleus

ISSN 1650-2884

LU-CS-EX: 2024-10

DEPARTMENT OF COMPUTER SCIENCE

LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-10

Artikelplaceringssystem för industriell robot med constraintprogrammering

Astrid Dymling, Martin Nybleus

Artikelplaceringssystem för industriell robot med constraintprogrammering

Astrid Dymling
Astrid.Dymling@outlook.com

Martin Nybleus
Martin.Nybleus@gmail.com

27 februari 2024

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Olle Hedbrant, olle.hedbrant@cognibotics.com
Per Andersson, per.andersson@cs.lth.se

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Innehåll

Abstract, English

This thesis, commissioned by Cognibotics, covers the development and implementation of a placement system for a pick-and-place robot. The solution has also been implemented as a proof-of-concept on the HKM1800, manufactured by Cognibotics. The model takes into account various constraints to find a solution. The results are discussed against other methods in terms of quality of the solution, computation time, and other aspects. The system developed uses constraint programming to find a volume-optimal solution for a pre-known e-commerce order. The system uses computer vision to identify, at a picking location, picking positions for each package in a configuration and places items according to the placement pattern that guarantees volume optimality in most cases for the specific problem addressed by the work. The work shows that constraint programming is a viable approach, and perhaps the best one for this type of packing problem.

Abstract, svenska

Detta examensarbete, på uppdrag av Cognibotics, omfattar utvecklingen och implementeringen av ett placeringssystem med constraintprogramming. Lösningen har implementerats som proof-of-concept på Cognibotics pick-and-place-robot HKM1800. I systemet ingår en modell utvecklad i MiniZinc, ett verktyg och språk för constraintprorgammeringsbaserad modellering, som utifrån begränsningar från kravspecifikation tar fram placeringsmönster som garanterar volymoptimalitet. Systemet använder sig av datorseende för att vid en plockplats identifiera paketen och placerar ut artiklar enligt placeringsmönstret. Arbetet visar att constraintprogrammering är en gångbar metod, kanske den bästa för den här typen av packningsproblem.

Förord

Med detta arbete gör vi vårt sista akademiska avtryck här vid teknisk fysik på LTH och avslutar med stolthet en fantastisk tid full med utmaningar, lärdom och nöje.

Vi vill rikta ett stort tack till våra vänner och kollegor på Cognibotics som med stort engagemang och härlig energi gjort de senaste månaderna till en rolig och lärorik tid för oss.

Ett speciellt tack vill vi ge till vår handledare på företaget, Olle Hedbrant, för ditt värdefulla stöd och vägledning genom hela processen.

Tack till Klas för dina idéer, stöd och förtroende.

Tack till Per Andersson, vår handledare på LTH, för att ha hjälpt oss forma arbetet och styra det i rätt riktning.

Tack Johan Lundgren för all hjälp med visionsystemet och dina kloka insikter generellt i arbetet. Tack till Sven G. Robertz och Mattias Wallinius för hjälp med C-programmering. Tack hela HKM-gänget för alla skratt.

Vi är djupt tacksamma för alla som har bidragit till detta projekt och betonar deras betydelse för vårt arbete. Utan er hade detta arbete inte varit möjligt.

Arbetsfördelning

Under förstudien tog Dymling ett större ansvar för att utforska annan programvara medan Nybleus ansvarade mer för att utforska relaterade arbeten.

Utvecklingen och testningen av constraintlösaren genomfördes huvudsakligen av Dymling medan Nybleus skötte det mesta rörande integrationen av constraintlösaren med robotens PLC.

Visualiseringsverktyget som användes för att verifiera lösningar har utvecklats av Dymling. Nybleus implementerade C-kod med hjälp av Mattias Wallinius och Sven Robertz. Dymling ansvarade för utveckling av ST-script, understött av handledare Olle Hedbrant och Johan Lindberg. Nybleus ansvarade i större utsträckning för Keba-delen av systemet och Dymling för visiondelen.

Testning av artikelplaceringssystemet på HKM gjordes tillsammans, bland annat eftersom det krävdes flera personer för att göras effektivt.

Kapitel 1 är huvudsakligen skrivet av Nybleus. Kapitel 2 är skrivet av Dymling, undantaget avsnitt 2.1.1, 2.3 samt 2.5.5 som är skrivet av Nybleus. Avsnitt 3.1 är skrivet av Dymling och 3.2 av Nybleus. Kapitel 4 och 5 är i stort sett skrivet av Nybleus med undantag för avsnitt 4.1 som är skrivet av Dymling.

Figurer

Tabeller

Modeller

Kapitel 1

Inledning

1.1 Bakgrund

1.1.1 Cognibotics AB

Cognibotics är ett svenskt företag som genom sin forskning inom robotik strävar efter att bidra till samhällets industriella utveckling med nya automationslösningar och riktar sig bland annat mot kunder som vill effektivisera sina industriprocesser [3]. Detta examensarbete, på uppdrag av Cognibotics, kommer att utgå ifrån ett case gällande en av deras kunder med ett sådant behov. För Cognibotics uppstår packningsproblem i många av pick-and-place roboten HKM:s (Hybrid Kinematic Machine) användningsområden. Detta arbete utgår alltså från HKM:s situation, men delar av lösningen kan appliceras på andra användningsområden. Innan detta arbete hade HKM i princip ingen placementlogik mer än att den slumpmässigt placerar artiklar för att de inte ska staplas på varann, se figur 1.1.

Detta arbete utgår ifrån ett specifikt placeringsproblem inom e-handel som Cognibotics fått i uppdrag att lösa av kund, men utforskar även andra områden. Det finns olika metoder för att lösa olika typer av packningsproblem och i det här arbetet utnyttjas constraint programming (CP) av anledningar som beskrivs längre fram i rapporten. Vidare finns det i dagsläget få, om ens några, vedertagna metoder för industriella robotar att praktiskt placera konfigurationer av artiklar [1]. Examensarbetet ämnar att utforska placeringsmetoder för industriell robotik

[1] Källa: Mats Jonsson, affärsenhetsansvarig på Cognibotics och Industriexpert

Figur 1.1: Användningen av HKM i samband med en plockstation där roboten, enligt den tidigare placeringslogiken, slumpmässigt distribuerar objekt inom det angivna placeringsområdet.

samt ta fram och implementera en egen lösning med hjälp av constraintprogrammering som löser packningsproblemet både i teorin och praktiken.

1.1.2 Frakt av tomt utrymme

Packningsproblem är centralt inom logistik, där målet är att optimera utnyttjandet av utrymmet i olika typer av behållare eller containrar [7]. Problemet uppstår när man vill packa en samling föremål med varierande storlekar inom begränsade behållare eller ytor så att utrymmet används på ett så effektivt sätt som möjligt [12]. Packningsproblem är centralt inom många logistiska aktiviteter, som transport och lagerhantering [7]. I litteraturen benämns variationer av problemet ofta som *Container loading* eller *3D bin packing (3DBP)*. Dessa problem är av typen NP-hard *(Non Polynomial Time Hard)* och i många fall tidskrävande att lösa exakt, varför inexakta och icke-optimala lösningsmetoder ofta används [12] [18].

Förbränning av fossila bränslen står för den största delen av utsläppen i Sverige och transportsektorn bär ett ansvar i att göra vad som går för att minimera dessa [13]. Att minimera frakten av tomt utrymme och således uppnå en högre nyttjandegrad vid transport bidrar till ett färre antal transporter, vilket gör frågeställningen detta arbete undersöker högst aktuell inte minst ur ett miljöperspektiv. Tomt utrymme syftar på både luft men även användning av luftkuddar eller andra utfyllnadsmaterial. Europakommissionen förbereder en ny lagstift-

ning som innefattar åtgärder för att begränsa frakten av tomt utrymme. I det nuvarande förslaget skall andelen tomt utrymme begränsas till 40 % i flera typer av frakter, inklusive e-handelspaket [6]. Denna nya lagstiftning kommer att få konsekvenser för företag som är beroende av e-handel. Företag kommer att behöva anpassa sina förpackningsmetoder för att följa de nya reglerna. För Cognibotics skulle det innebär det att de gentemot sina kunder måste kunna säkerställa att HKM packar med max 40 % tomt utrymme.

1.2 Packningsproblem A - Fallstudie

Cognibotics mål för HKM kommer på sikt vara att lösa flera typer av packningsproblem åt företag som önskar automatisera sin lagerhantering. Det här arbetet och den här rapporten kommer att utgå från ett specifikt packningsproblem inom E-handel, som vi kommer referera till som Packningsproblem A (P:A). Problemet har Cognibotics tagit på sig i uppdrag att lösa åt en kund och är specifikt till just deras situation. Det är det problemet som vi, författarna, fått i uppdrag att undersöka och lösa.

1.2.1 Förutsättningar

HKM ska packa ordrar där de ingående artiklarna är kända på förhand. Sortimentet består av endast 11 artiklar men dessa kan packas i vilka konfigurationer som helst, även om vissa är vanligare än andra. Oftast rör det ett litet antal paket, upp till 7 stycken i 99 procent av fallen enligt data från Cognibotics kund. Snittbeställningen ligger på endast 2,5 artiklar.

Då kundordern kommer in tillåts Cognibotics system att välja i vilken ordning de ska skickas till HKM. Ordningen på artiklarna är alltså känd. Sedan ställer HKM artiklarna i ett mönster, exempelvis enligt detta arbetes lösning. Efter att artiklarna placerats så bandas de tillfälligt innan de skickas in en packningsmaskin. Denna packningsmaskin anpassar paketeringen efter måtten på konfigurationen som skickas in, givet att pakethögen håller sig inom packningsområdets maxgränser vad gäller längd, bredd, och höjd på 600x400x300mm. Därav är syftet med packningsalgoritmen att ställa objekten på sådant sätt att det omslutande rätblocket blir så litet som möjligt till volymen [2].

1.3 Syfte

Syftet med detta examensarbete är att presentera en lösning för P:A genom CP, samt att utvärdera lösningen och betrakta andra metoder. Arbetet ämnar att svara på frågan om CP är

[2]Källa: Mats Jonsson, affärsenhetsansvarig på Cognibotics och Industriexpert

lämpligt för att lösa place-problemet inom e-handel, med utgångspunkt i P:A. Vidare kommer en implementation genomföras som ämnar att svara på frågor kring vilka svårigheter som finns i att omsätta en teoretisk placering till ett fungerande flexibelt system för HKM. Arbetet kommer också att resultera i andra insikter och nyttig information för Cognibotics och framtida arbeten.

1.3.1 Forskningsfrågor

De forsknignsfrågor rapporten avser att besvara är

- Vad finns det för existerande lösningar till 3D-placeringsproblem?
- Är constraintprogrammering en lämplig metod för att lösa 3D-placeringsproblem?
- Vilka svårigheter finns i att omsätta en teoretisk placering till ett fungerande system för HKM?

1.3.2 Avgränsningar

Alla paket approximeras även till rätblock. I övrigt har det teoretiska placeringsproblemet avgränsats till specifikationerna som beskrivs i fallstudien.

Vidare är den praktiska implementationen på en proof-of-concept-nivå och behöver vidareutvecklas för att nå kommersiell gångbarhet.

1.4 Relaterade arbeten

Till vår kännedom finns idag få, om några, effektiva system som har implementeras på pick-and-place-robotar för att hantera den här typen av placeringsproblem. Placeringsproblem verkar dessutom vara tillräckligt varierade och komplexa för att det inte ska finnas någon ”once size fits all”-metod.

Den här rapporten innefattar en förstudie där relaterade arbeten undersökts. Ett av dessa genomfördes av Martello et al. som utvecklat en exakt algoritm för en variant av 3DBP-problemet [12]. Deras förutsättningar skiljer sig dock från de i detta arbete eftersom de packar i flera förutbestämda bins/behållare medan systemet som beskrivs i detta mastersarbete optimerar över en enda behållare, där rätblocken också tillåts att rotera, vilket Martellos inte gör. De lyckas lösa sitt problem optimalt för upp till 30 paket.

Andra relaterade arbeten är exempelvis de som rör online bin packing, det vill säga när paketen kommer på löpande band, däribland arbeten av Zhao et al. och lopez et al. [11] [18]. Båda dessa använder sig av maskininlärningsbaserade (ML-baserade) lösningsmetoder. Detta problem är inte heller det som i huvudsak rörs av denna rapport.

Lopez et al. genomförde en fingervisande jämförelse för ett mellan en sådan ML-baserad lösning för online bin packing och en optimal/exakt lösning och en inexakt algoritm. I experimentet packades upp till 19 paket vars alla dimensioner är 1, 2 eller 3 i en volym om 5x5x5. Den optimala lösningen utnyttjar ca 10 procent mer utrymme än deras RL-modell och ca 20 procent mer utrymme än den inexakta algoritmen, se figur 1.2 för data för alla 3 experiment [11].

	Packing Density			Of Max Packing Density		
Experiment	Optimisation	DRL	Rules Based	Optimisation	DRL	Rules Based
1	93.5%	83.2%	75%	100%	89%	80%
2	88.35%	80.4%	74.8%	100%	91%	84%
3	93.6%	80.5%	71.8%	100%	86%	77%

Figur 1.2: Tabell från Lopez et al. som jämför exakta och inexakta algoritmer med deras ML-lösning för deras packningsproblem.

1.5 Förslag på alternativa lösningar från Cognibotics

Cognibotics kom med två förslag på programvara som de misstänkte skulle kunna lösa problemet. I detta avsnitt går vi kort igenom de programmens möjligheter.

1.5.1 Program 1: Black Box

I samtal med säljrepresentant för program 1 vill inte denne redogöra för den bakomliggande tekniken i programvaran som därav betraktas som en black box. Programvaran är utvecklad för att hitta placeringar till problem med betydligt fler ingående paket än P:A, med möjlighet att välja flera och stora containers. Exempel på sådana problem återfinns i frakt med lastfartyg eller lastbilar. De kvalitativa fördelarna med denna programvara är att den är beprövad av flera företag och har funnits länge på marknaden. Den tillåter också flexibilitet i att kunna definiera tillåtna rotationer för paket. med mera. Dock finns det inget sätt att exakt modellera ett problem som P:A i denna programvaran. Detta eftersom man i den måste välja storleken på sina behållare på förhand, medan målet i P:A är att hitta vilken den minsta teoretiska behållaren är.

1.5.2 Program 2: ML- och visionbaserad placering

Program 2 erbjuds av samma företag som tillhandahåller HKM:s nya visionsystem som implementeras samtidigt som detta arbete, fast på andra robotar. Systemet är egentligen utvecklat till annan situation än P:A. I den situationen är alla objekt som ska placeras inte är kända på förhand, utan bara en i taget. Det gör att man inte kan garantera en optimal lösning av problemet på samma sätt som om man på förhand känner till artiklarna. Dessutom är detta system i sin nuvarande version aningen begränsat och placerar bara artiklar där den ”ser” att det finns plats.

Kapitel 2

Metodik

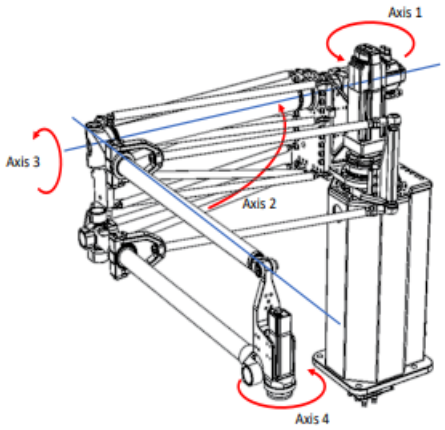
2.1 HKM1800

HKM1800 är en pick and place-robot av serie- och parallellhybridkinematisk arkitektur utvecklad av Cognibotics. Med en radiell räckvidd på 1800 mm kan robotarmen hantera förhållandevis långa avstånd och många positioner över en stor yta. Då robotarmen består av lättviktiga komponenter i armen är upp till 80 % av manipulatorns vikt koncentrerad i den stationära basen, vilket gör att roboten kan hålla ett relativt snabbt arbetstempo [4].

Roboten är utrustad med aktiv rörelseåterkoppning från handleden för att uppnå jämna och precisa rörelser. Vid rörelse kompenserar systemets reglertekniska mekanism för de variationer i belastning och avvikelser som uppstår. För att plocka och placera objekt använder HKM sig av olika greppdon som klickas fast till armen kombinerat med en vakuummekanism längst ut på greppdonet. Olika greppdon kan användas för att passa det specifika objektets form och vikt. Cognibotics utvecklar den reglertekniska programvara som styr HKM:s kimenatik och även annan typ av programvara som exempelvis pick-and place-system [1].

I figur 2.1 nedan syns en illustration av HKM1800 och dess fyra rörelseaxlar. Axel 1 tillåter en rotation på $\pm 180^{o}$, axel 2 $\pm 56^{o}$, axel 3 $\pm 32^{o}$ och axel 4 är obegränsad. Dessa frihetsgrader ställer krav på artiklars orientering vid plock så väl som sätter begränsningar vid placering och diskuteras vidare i avsnitt 5.1.

[1] Källa: Mats Jonsson, affärsenhetsansvarig på Cognibotics och Industriexpert



Figur 2.1: Skiss av HKM1800 med rotationsaxlarna inkluderade.

2.1.1 Programmable Logic Controller (PLC)

Programmable Logic Controller (PLC) är benämningen på den datorenhet som används för att styra HKM och många andra robotar. Som de flesta PLC-enheter programmeras denna i språket Structured Text (ST) [17]. Koden exekveras cykliskt med en cykeltid på 1 ms. Cyklisk sekvensering innebär att all kod i programmet körs varje cykel. Dock kan man variera vilka block eller satser som körs mellan cyklerna, eftersom det styrs av systemets variabler som sparas däremellan. Detta styrs bland annat genom ”State machines”. En State Machine används för att modellera och styra system med flera tillstånd eller lägen. Den består av ett antal tillstånd och övergångar mellan dessa tillstånd baserat på specifika händelser eller villkor. Eftersom PLC:ns kod exekveras cykliskt krävs en state machine för att att hålla koll på var i ett händelseförlopp roboten befinner sig, t.ex. startläge, plockläge, placera-läge och stoppläge.

Varje cykel i sekvensen kan alltså inkludera rörelser, kameraavläsningar, beslut och andra aktiviteter som krävs för att utföra en specifik operation, vilket sedan påverkar beslut och rörelser i cykeln efter. Genom att använda cyklisk sekvensering kan roboten kontinuerligt utföra önskade uppgifter och repetera sekvensen enligt behov. Detta har ingen påverkan på packningsproblemet utan endast på den praktiska implementationen.

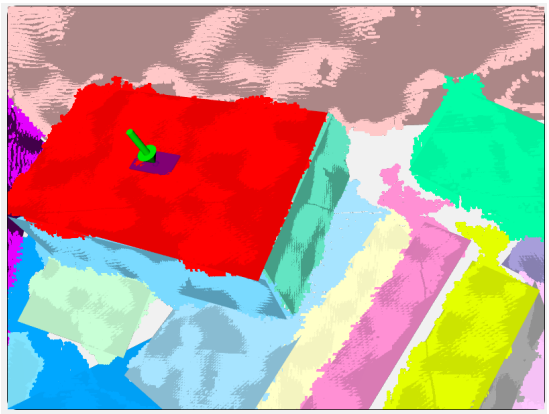
Keba är företaget som utvecklar programvaran som styr HKM:s PLC. Mjukvaran består av KeStudio som är den utvecklingsmiljö där script och program skrivs för att sedan laddas upp och köras på PLC:n. Även C++-kod kan kompileras, laddas upp, och köras på PLC:n genom Kebas eclipsebaserade editor CPPedit. C-funktionerna kan kallas på i skripten som är skrivna i Structured Text eller Kairo [9].

För att styra PLC:n och roboten används en teach pendant vilken fungerar som ett användargränssnitt mellan operatören och roboten. Teach pendanten består av en pekskärm och

dedikerade knappar för att styra roboten, samt nödstop och en dead-switch av säkerhetsskäl.

2.2 Vision

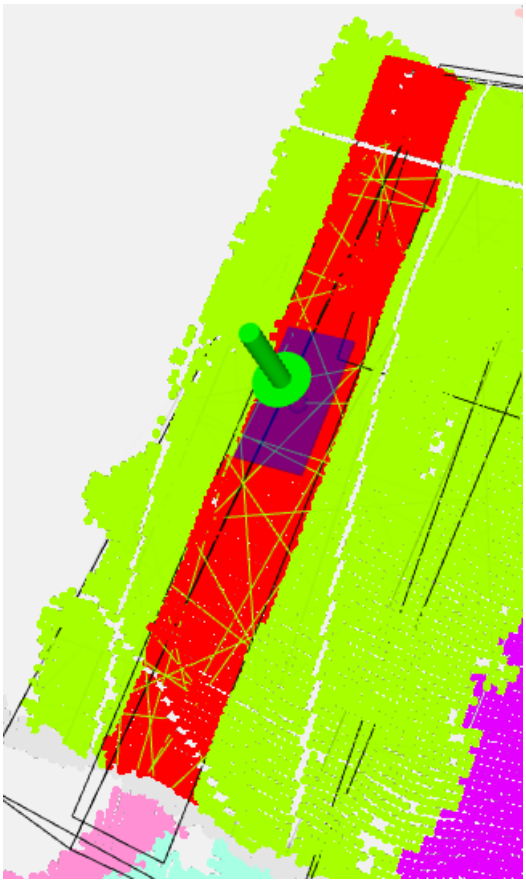
För att identifiera objekt och deras plockpositioner är användningen av ett visionsystem nödvändig. Cognibotics har nyligen övergått till ett nytt, mer avancerat system för deras kommande produkter och använder sig således inte längre av systemet som har använts i detta arbete. Detta system fungerar genom att ta en bild av ett objekt för att sedan hitta en optimal plockposition på objektets ovansida. Den identifierade plockpositionen ligger vanligtvis i mitten av den yta som var synlig för kameran och HKM:s plockverktyg.
Visionsystemet returnerar bland annat en vektor som är parallell med objektets längsta riktning. Från början var denna funktion endast tänkt att användas för avlånga plockverktyg som behöver orientera sig åt samma håll som ytan som skall plockas, se figur 2.2. För implementeringen av placeringssystemet är denna data fördelaktig då ett objekts rotation är av nödvändig att beakta för att placeringen ska bli korrekt. Utan detta måste man veta objektens rotation på förhand. I figur 2.3 syns en bild på ett större, rektangulärt objekt där även visionsystemets djupseende illustreras.



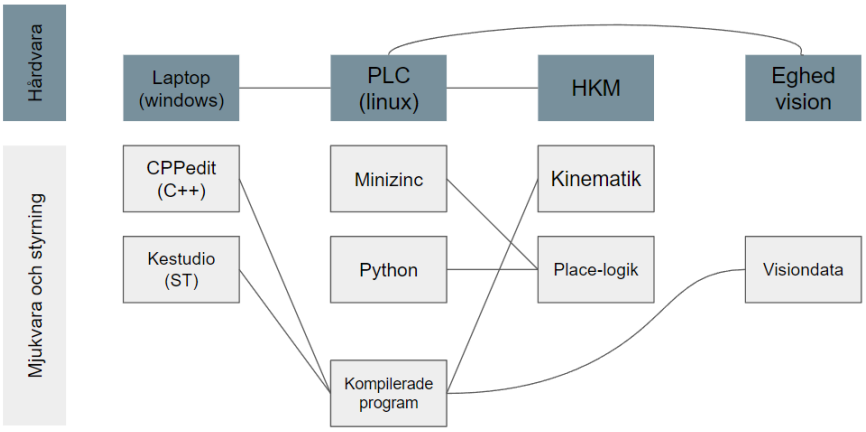
Figur 2.3: Större rektangulärt objekt i visionsystemet. Det röda representerar det avlånga objektet. Den gröna pilen pekar på plockpositionen. Det lila området representerar objektets yta.

2.3 Integration och implementering

Många delsystem samverkar alltså för att HKM ska kunna plocka och placera objekt. Med delsystem så menas HKM, placementlogiken från minizinc, vision, och så vidare. En överblick över delarna i implementationen finns i figur 2.4.



Figur 2.2: Avlångt objekt i visionsystemet. Det röda representerar det avlånga objektet. Den gröna pilen pekar på plockpositionen. Det lila området representerar objektets yta.



Figur 2.4: En kvalitativ överblick över delarna i implementationen. Linjerna ger en fingervisning om de delar som är sammakopplade.

Delarna som rörs av det här arbetet är framförallt place-logiken och viss kinematik (exempelvis rotation) kopplat till det, några funktioner inklusive en state machine skriven i C++ för att kunna läsa in place-lokigen, samt huvudscriptet. Värt att notera är att utdaterade versioner av python och minizinc körs, 2.7 respektive 2.1, för att vara kompatibla med linux debian som körs av PLC:n.

En state machine har implementerats i syfte att köra minizincprogrammet och läsa in datan. Den är skriven i C++ och kallas på från huvudskriptet i PLCn. Den består av 5 states som tillåter beräkningarna och inläsningen att ske under PLC:ns cykliska sekvensering av koden. Grunden till koden för inläsning skrevs av Sven G. Robertz. Vi fick även hjälp av Mattias Wallinius med att programmera grunden till vår state machine i C++.

2.4 Problemformulering: 3DBP-problemet

Problemet i arbetet går ut på att placera objekt av olika storlek i en begränsad utrymmesenhet på sätt som minimerar användningen av utrymme. Detta är känt som ett så kallat NP-hard problem och står för ”Non-deterministic Polynomial time Hard”, vilket innebär att det inte finns någon bevisad metod som löser det inom polynomiell tid, men en lösning kan verifieras på polynomiell tid [12] [5]. För att hitta en optimal lösning måste således alla kombinationer prövas, vilket kan vara tidskrävande för ett stort problem. Några vanligt förekommande variationer av packningsproblem är exempelvis 3D bin packing eller container loading [2]. Bin packing i 3D innebär att man ges ett antal ”bins”, eller behållare, som rymmer en viss volym, och ett antal paket av olika storlekar som skall passa i behållarna. Målet är att använda så få behållare som möjligt för att packa alla paket. Fallet med P:A kan ses som en variant av 3D

bin packing problem där de tillgängliga behållarna är ett set av alla rektangulära behållare med heltalsdimensioner upp till 600x400x300mm. Optimeringsproblemet ligger i att hitta den minsta behållaren där alla paket får plats.

Denna variant av 3DBP är definierat för ett förbestämt ändligt set $I = \{i_1, i_2, ..., i_n\}$ av objekt, approximerade som rätblock och ett set dimensioner $s(i) = \{s_x, s_y, s_z\}$, $s(i) \in \mathbb{Z}$ för varje $i \in I$. Rätblocken har längd l_i, bredd b_i och höjd h_i. Problemet är enligt formulering att hitta den behållare, även denna rätblocksformad, med dimensionerna $L > l_i$, $B > b_i$ och $H > h_i$ som erhåller den minsta volym där alla rätblock från I får plats. Alla rätblock från I får roteras. I de scenarion Cognibotics efterfrågar en lösning för (och inom e-commerce generellt) är antalet varor som ska packas relativt få, under 10 st per order i detta fall. Vilka varor som ska packas är dessutom känt på förhand. Det talar för att en lösning baserad på *constraintprogrammering* skulle vara lämplig och kan uppnås inom en rimlig tidsram.

2.5 Constraintprogrammering

Constraintprogrammering (CP) är ett verktyg som kan användas till att hantera och lösa kombinatoriska sökproblem. CP tillämpas inom en mängd olika områden, så som t.ex. packningsproblem, schemaläggning och vehicle routing. Användaren kan uttrycka ett verkligt problem som dessa i termer av *beslutsvariabler (eng: decision variables)* och *constraints* varpå en *constraintlösare* (eng: *constraint solver*) tilldelar värden till beslutsvariablerna med hänsyn till definierade constraints [1]. En av fördelarna med CP är att det stödjer en deklarativ programmeringsform, vilket innebär just att man kan fokusera på de krav en önskad lösning ska uppfylla, snarare än att utveckla algoritmer för att uppnå dem. [15].

En svårighet med att använda sig av constraintprogrammering för att lösa problem är att hitta unika lösningar till problemet. Beroende på problemets formulering kan det uppstå många så kallade symmetriska lösningar. Detta bidrar till en längre lösningstid och i värsta fall att den optimala lösningen inte hittas. Ett sätt att göra sig av med symmetriska lösningar är att införa så kallade *symmetry breaking constraints*, alltså symmetribrytande constraints [10].

2.5.1 MiniZinc

Det finns inga standardspråk eller standardmetoder för constraintprogrammeringsbaserad modellering. I det här arbetet används MiniZinc som är ett open source modelleringsspråk och -verktyg, utvecklat just för lösa problem med constraintprogrammering [14]. I MiniZinc

ges möjligheten att beskriva problemets begränsningar och låta en solver hitta lösningar inom dessa begränsningar. Minizincmodellerna är små och kan köras i många miljöer. I vårt fall är det till och med möjligt att köra minizinc på robotens operativsystem.

2.5.2 Utveckling av modell

För att programmera en constraint-lösare behöver användaren definiera följande komponenter som utgör modellen:

1. Beslutsvariabler och deras *domäner*,
2. Kostnadsvariabel, om optimering önskas,
3. Constraints, samt
4. Sökmetod. [1]

2.5.3 Variabler och domäner

För att uttrycka P:A i MiniZinc-modellen (MZM) krävs att variablerna och deras domäner definieras. Eftersom vi i P:A efterfrågar en lösning på hur vi som mest platseffektivt placerar en uppsättning paket kommer en av beslutsvariablerna innehålla placeringspositioner. Domänerna definierar ett giltigt intervall av numeriska värden för varje variabel som i detta fall är begränsningarna för robotens rörelseområde samt placeringsytans gränser. Optimeringsproblem kräver att man definierar ytterligare en beslutsvariabel som sökningen ska optimera utifrån. Detta är känt som lösningens *kostnadsfunktion* eller *kostnadsvariabel* vars värde ska minimeras.

2.5.4 Constraints

Constraints är de begränsningar som specificerar vilka kombinationer av värden som är giltiga för variablerna. I P:A skulle dessa begränsningar kunna inkludera regler för att undvika kollisioner mellan paketen, begränsningar för objektens placering baserat på deras egenskaper och regler för robotens rörelsebegränsningar. Genom att definiera och hantera dessa begränsningar kan CP hitta lösningar som uppfyller alla krav och begränsningar.

2.5.5 Integration med MiniZinc och FlatZinc

För att lösa problem modellerade i MiniZinc krävs en lösningsmotor som kan utföra sökning och optimering. Det finns flera lösningsmotorer som kan användas med MiniZinc, inklusive Gecode och JaCop [16][8]. Genom att välja en lämplig lösningsmotor kan man dra nytta av dess specifika egenskaper och prestanda. Gecode är utvecklat i C++ och är en av de lösningsmotorer som kan integreras med MiniZinc. I detta arbete används Gecode eftersom den lösaren är kompatibel med PLC:ns operativsystem.

Integrationen av Gecode med MiniZinc möjliggör användningen av Gecodes sökalgoritmer och optimeringsfunktioner som en ”motor” för att lösa problem som är modellerade i MiniZinc. MiniZinc tillhandahåller ett mellanformat kallat FlatZinc som används för att representera MiniZinc-modeller på ett standardiserat sätt. Denna representation kan sedan tolkas av Gecodes interpreter för att lösa problemet. FlatZinc är en minimalistisk representation av problemet och består av en lista med variabler, domäner och constraints. När en MiniZinc-modell kompileras av MiniZinc-kompilatorn genereras en FlatZinc-representation av modellen, som sedan kan tolkas av Gecodes interpreter. Gecodes interpreter läser in FlatZinc-modellen och översätter den till ett internt representerbart format för att lösa problemet. Interpretern använder sedan sina sökalgoritmer och optimeringsfunktioner för att hitta lösningar inom de definierade begränsningarna.

2.6 Utförande

Examensarbetet genomfördes med en kombination av praktisk implementering och teoretiska studier. Arbetet innefattar en utvärdering av modellen samt systemet i sin helhet genom simuleringar och testningar på HKM.

2.6.1 Case

För att utveckla och implementera systemet har vi utgått ifrån det nämnda fallet inom e-commerce som involverar ordrar av ett relativt litet antal paket. Roboten ska plocka upp ett paket från en plockyta där en bild kommer att tas för att identifiera paketet. Sedan ska det placeras på en optimal plats i högen på placeringsytan så att högen är stabil. Vi behöver således känna till den optimala platsen för varje paket. Därför börjar vi med att lösa packningsproblemet genom att utveckla en algoritm som baserat på order- och artikeldata som input, genererar 3D-placeringsmönster. Detta görs med en constraint solver modell i MiniZinc.

2.6.2 En första modell

Utvecklingen av modellen inleddes med att skapa en simpel modell där input och variabler definieras samt ett enda constraint införs, att paketen inte får överlappa. I den här modellen nöjde vi oss med att hitta en lösning, vilken som helst, som tillfredsställer överlappningskravet. Efter att ha testkört programmet och uppnått en lösning inom en acceptabel tid för ett litet antal paket infördes en kostfunktion för att optimera totalvolymen av kuben som pakethögen spänner upp där lösningen som minimerar värdet av denna fås. Modell 1 är denna simpla modell i pseudokod.

Modell 1: Simpel modell med överlappningsbegränsning och minimering med avseende på totalvolym.

1 **Inmatning:** Antal kuber n, kubpositioner box_posn, kubmått $boxes$;
2 **Begränsning:** Säkerställ att kuber inte överlappar. För detta används diffn_k. ett constraint i minizinc.;
3 $constraint\ diffn\_k(box\_posn, boxes)$;
4 **Beräkna kubernas utbredning - dvs deras position plus storlek i 3 dimensioner:**;
5 *# Beräkna utbredning i x-led*;
6 $box\_posnx \leftarrow \text{column}(box\_posn, 1)$;
7 $posPLUSsizeX[i] \leftarrow box\_posnx[i] + boxes[i, 1]$;
8 *# Beräkna utbredning i y-led*;
9 $box\_posny \leftarrow \text{column}(box\_posn, 2)$;
10 $posPLUSsizeY[i] \leftarrow box\_posny[i] + boxes[i, 2]$;
11 *# Beräkna utbredning i z-led*;
12 $box\_posnz \leftarrow \text{column}(box\_posn, 3)$;
13 $posPLUSsizeZ[i] \leftarrow box\_posnz[i] + boxes[i, 3]$;
14 **Sök optimal placering:**;
15 *# Beräkna totalt begränsande volym*;
16 $volume \leftarrow \max(posPLUSsizeX) \times \max(posPLUSsizeY) \times \max(posPLUSsizeZ)$;
17 *# Lös för minsta volym*;
18 **minimera** $volume$;

Härifrån skedde utvecklingen av modellen genom att successivt öka problemets komplexitet, utvärdera modellens prestanda och utveckla modellen. Utvärderingen skedde i två steg. I första steget kontrollerades huruvida en lösning hittades inom en rimlig tid. I nästa steg, ifall en lösning hittades, granskades den för att se så att krav så som stabilitet uppfylldes. Komplexi-

tetsnivån ökades genom att successivt öka antalet paket eller genom att öka antalet dubletter av en artikel men behålla totala antalet paket. När modellen inte fann en lösning eller inte fann en lösning inom en rimlig tid infördes nästa constraint för att skära ned sökrummet ytterligare. För att utvärdera lösningarna under utvecklingen av modellen användes ett användes ett internt utvecklat visualiseringsverktyg.

2.6.3 Indata och parametrar

Programmet läser in en datafil som hämtats från e-handelsordrar i ett överordnat system hos Cognibotics kund. Datafilen innehåller information om antalet artiklar som ska ingå i en konfiguration, representerat av parametern `n`, samt artiklarnas dimensioner som representeras av en array `boxes` av dimension 1xn bestående av n stycken set där varje set innehåller tre element som representerar en artikels mått i x-, y- och z-led.

2.6.4 Variabler

Tabell 2.1 visar de ingående variablerna i programmet där `n` och `boxes` hålls konstanta enligt beskrivna värden i tidigare avsnitt. För att möjliggöra att paketen kan rotera har en 2D array `boxes_dim` av dimension nx3 introducerats bland variablerna där varje rad i representerar en artikel och kolumnerna representerar x, y och z-dimension. De fixa värdena från `boxes` får nu anta fri plats i x, y och z i `boxes_dim` som representeras av första, andra och tredje kolumnen i matrisen.

Namn	Beskrivning
box_posn	nx3 matris av int som är paketens positioner
boxes_dim	nx3 matris som anger paketens dimensioner och utbredningsriktning
boxes_marg	nx3 matris med paketens dimensioner och utbredningsriktning med pålagda marginaler
th	en int som anger minsta bredden ett pakets sida måste ha för att kunna stå på högkant
bigbox_index	int som håller koll på vilket index i boxes som motsvarar det största paketet
check_doubles	nxn matris av bool som anger förekomsten av en artikel
box_posnx	1xn matris med alla boxars x-position
box_posny	1xn matris med alla boxars y-position
box_posnz	1xn matris med alla boxars z-position
posPLUSsizeX	1xn matris med alla boxars utbredning i x-led
posPLUSsizeY	1xn matris med alla boxars utbredning i y-led
posPLUSsizeZ	1xn matris med alla boxars utbredning i z-led
biggest_dim	1xn matris med varje pakets största dimension
volume	totalvolymen av den kub som pakethögen spänner upp

Tabell 2.1: Tabell med alla ingående variabler i MiniZinc-skriptet.

2.6.5 Överlappning

`diffnk()` är ett globalt constraint i MiniZinc som i denna algoritm nyttjas som constraint för att begränsa k-dimensionella lådor så att de inte överlappar varandra. För varje låda i och dimension j, är `box_posn[i, j]` lådans basposition i dimension `j`, och `boxes_dim[i, j]` är storleken i den dimensionen. Se exempel på hur det kan användas i modell 1, rad 3.

2.6.6 Bryta symmetrier

Med utgångspunkt i P:A som tillåter programvaran att välja ordningen paketen kommer i, kan detta utnyttjas för att bryta symmetrier. Precis som en människa skulle stapla paket, tycks det vara intuitivt fördelaktigt att börja med det största paketet i en order och fortsätta stapla resten av paketen ovanpå.

Att låta det största rätblocket placeras först är därför ett sätt att eliminera symmetriska lösningar. `max()` returnerar det största elementet i en array och används således för att hitta största utbredningen i varje dimension. Vi undersöker raderna i `boxes_marg` och utnyttjar max() för att hitta största dimensionen för varje rad (paket). Dessa lagras i en array av

dimension 1xn som vi kallar `biggest_dim`. `arg_max()` returnerar indexet för det största elementet i en array. Vi inför ett nytt constraint som identifierar paketet med störst dimension vars index lagras i variabeln `bigbox_index` och placerar detta paket i origo.

En konfiguration som innehåller dubletter av en artikel kommer bidra till ett ökat antal symmetriska lösningar. För att kunna eliminera symmetriska lösningar som en konsekvens av dublett-artiklar låter vi algoritmen identifiera dubletterna och införa constraints, t.ex. i form av sortering. Vi definierar en matris `check_doubles` som innehåller ettor och nollor baserat på om artikel i har dubletter eller inte.

Ett exempel med 6 artiklar varav rätblock 1 och 3 respektive 4 och 6 är av samma artikel är

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

För att sätta ett symmetribrytande constraint som begränsar valmöjligheten av de många olika konfigurationer som dubletter medför bestämmer vi att de måste följa en viss ordning. Vi itererar därför igenom `check_doubles` samt matrisen med rätblockens positioner `box_posn` och kräver att summan av positionskoordinaterna i x-, y- och z-led för rätblock i vara mindre än eller lika med koordinatsumman för rätblock j, där vi har dubletter, dvs då `check_doubles[i,j]` = 1 och i ≤ j.

2.6.7 Fysikalisk kompabilitet

För att säkerställa en konfigurations stabilitet bör man titta på paketens stapelbarhet. Av denna anledning har ett constraint införts som tvingar paketen att ligga ner ifall de är smalare än en lämpligt vald tröskelbredd. I P:A finns till exempel den vanligt förekommande manualen som enbart är någon millimeter bred. Detta constraint säkerställer att för alla boxar `i` som har någon dimension `j` ≤ `th` (`th` har valts till 20 mm initialt) måste `box_posn[i,3]=j`.

För att ytterligare säkerställa konfigurationens stabilitet bör det finnas ett constraint som säger att ett pakets tyngdpunkt (mittpunkten eftersom det är rätblock) inte får ligga utanför paketet under.

2.6.8 Marginaler

En lådas dimensioner från artikeldata kanske inte alltid stämmer överens exakt med de verkliga dimensionerna. Lika så kan det skilja sig ytterligare i inputen från visionsystemet, varför dessa bör tas hänsyn till på något sätt för att undvika att systemet placerar paketen för tätt och riskerar att de slår i varandra. En lösning på detta är att låta lösaren optimera på dimensioner som är aningen större än rätblockens förutsatta dimensioner. Därför definierar vi en ny matris `boxes_marg` som adderar en lämpligt vald felmarginal till den ursprungliga dimensionsmatrisen `boxes_dim`.

2.6.9 Minimering av volym

Vi är intresserade av att hitta minsta volymen och låter därför denna vara en variabel som solvern minimerar. Constraintlösaren optimerar alltså utifrån volymen av en packning, dvs produkten av maximala utbredningen i x-, y- och z-led. Det innebär att lösningen som kommer fås är den med minst totalvolym utifrån de constraints som definierats. Variablerna `box_posnx`, `box_posny` och `box_posnz` innehåller alla rätblocks x-, y- respektive z-positioner. Dessa nyttjas för att sedan beräkna den största utbredningen i varje riktning. Eftersom positionen utgår från rätblockets bas adderar vi varje pakets mått i respektive dimension till dessa variabler och deklarerar nya variabler `posPLUSsizeX`, `posPLUSsizeY` respektive `posPLUSsizeZ` som arrayer av dimension 1xn. Kostnadsfunktionen `volume` är således produkten av `max(posPLUSsizeX)`, `max(posPLUSsizeY)` och `max(posPLUSsizeZ)`.

2.6.10 Artikelordning

För att minimera risken att robotarmen välter paket på sin väg tillbaka till plockstationen så byggs systemet så att paketen placeras enligt en viss ordning. Placeringssekvensen är sorterad på stigande z-koordinat i första hand, stigande y-koordinat i andra hand och sist med avseende på stigande x-koordinat. Alltså placeras objektet i origo först, därefter fylls staplingen på längs y-axeln och sedan x-axeln.

2.7 Rotation

För att kunna placera paketen korrekt oberoende av vilken orientering de har vid plocktillfälle behöver vi veta om ett paket kommer vara x- eller y-orienterat i placeringsytan. Vi behöver identifiera paketets orientering vid plocktillfället för att kunna skicka instruktioner till plockarmen att korrigera vridningen. Vi började med att definiera ett koordinatsystem

i plockytan som har samma orientering som placeringsytan. Felmarginalen i rotationen kan korrigeras genom att subtrahera eller addera en vridningsvinkel ϕ_v så att paketet hamnar parallellt med koordinatsystemets x- eller y-axel. Vi utnyttjade den av visionsystemet genererade vektorn $\bar{v}_l$ parallell med paketets längsta sida samt genererade en vektor parallell med ena axeln i omgivningens koordinatsystem $\bar{v}_k$. Med hjälp av definitionen av skalärprodukt kan vi få information om ϕ_v. Vinkeln mellan $\bar{v}_l$ och $\bar{v}_k$ är således

$$\phi_v = \arccos\left(\frac{\bar{v}_l \cdot \bar{v}_k}{||\bar{v}_l||||\bar{v}_k||}\right). \tag{2.1}$$

För att veta om paketet ska vara x- eller y-orienterat i placeringplanet extraherades placeringslösarens output (dvs dimension och koordinat) och lagrade en 1:a i PLC-koden ifall objektet har sin längsta dimension i x riktning och en 0:a ifall den har det i y-riktning. Ifall paketet är x-orienterat ska det vridas $90° - \phi_v$ och ifall det ska vara y-orienterat ska det vridas ϕ_v.

Beroende på åt vilket håll paketet är roterat med en vinkeln ϕ_v måste vi veta om paketet ska roteras medurs eller moturs för att hamna rätt i placeringsytan. Detta kan lösas genom att använda kryssprodukten mellan vektorerna från visionsystemet och huruvida paketet ska roteras medurs eller moturs bestäms av huruvida kryssprodukten pekar neråt eller uppåt.

Kapitel 3

Resultat

I det här kapitlet redovisas resultaten. De har delats upp mellan lösarens prestanda och en kvalitativ redogörelse för utfallet av placeringarna i praktiken.

3.1 Lösarens prestanda

Testerna har gjorts baserat på verklig orderdata från case-företaget och redovisas i tabell 3.1. För de konfigurationer som består av 4-10 artiklar har alla konfigurationer testats. För de konfigurationer med 2-3 ingående artiklar valdes 10 konfigurationer från vardera kategori ut slumpmässigt. Testarna utfördes genom att se om MZM kunde hitta den optimala lösningen inom 3000ms. Konfigurationen med 6 artiklar som misslyckades innehöll 6 stycken likadana artiklar. Konfigurationen med 7 artiklar som misslyckades innehöll en dublett. För medeltiden exkluderades lösningarna som inte hittades optimalt på en given maxtid (3000 ms). I fallen när en optimal lösning inte hittades inom maxtiden så fanns i samtliga fall en lösning som höll sig inom maxgränserna för packningsmaskinen i fallstudien P:A.

nArtiklar	nKonfig	optimalt lösta	optimalt lösta %	medeltid (ms)
2	10	10	100	84
3	10	10	100	87
4	10	10	100	121
5	4	4	100	94
6	4	3	75	98
7	2	1	50	124
10	1	0	0	n/a

Tabell 3.1: Resultat från testerna för olika konfigurationer med varierande antal artiklar.

3.2 Placering

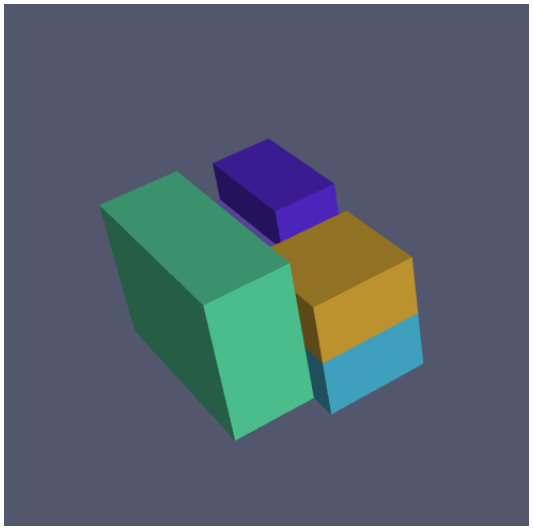
I figur 3.1 till 3.3 presenteras packningar som resultat av det implementerade systemet. I figur 3.4 presenteras en visualisering av MZM:s framtagna lösning för konfigurationen i figur 3.1. Tiden för ett plock varierar men kan upp till 1 minut per paket. Detta för att en bild ska tas och analyseras för att hitta plockpositionen. Detta tar ca 15-30 sekunder. Därefter kördes roboten i långsam hastighet av säkerhetsskäl och plockade upp paketet för att sedan placera det enligt MZM:s koordinater.

(a) **(b)**

Figur 3.1: Fem paket som placerats. Se visualisering i figur 3.4.

Figur 3.2: Kanter som inte placerats helt räta mot varandra.

Figur 3.3: Volymoptimal placering av konfiguration innehållande tre godtyckligt utvalda artiklar.



Figur 3.4: Visualisering av placeringslösarens output för packningen i figur 3.1.

Kapitel 4

Diskussion

4.1 Implementeringen

Visualiseringen i figur 3.4 visar att packningen följer det önskade mönstret och att paketen är korrekt positionerade. Systemet hanterar olika paketuppsättningar enligt problembeskrivningen. Implementering blev enligt planen ett proof-of-concept och en del problem återstår att lösa innan systemet är produktionsredo. De flesta av dessa grundar sig i att visionsystemet som användes har en del fel. Tiden det tar för bildtagning och bildbearbetning är väldigt lång och står för en stor del av tiden som ett helt plock- och placeringsmoment tar. Att integrera ett annat visionsystem i implementationen skulle därför potentiellt korta ner tiden för en packning avsevärt och bidra till den önskade effektiviteten.

Eftersom visionsystemets djupseende är opålitligt blev även vinkelberäkningarna för objektens rotationer felaktiga, varför rotationskorrigering inte implementerades. Detta fel konstaterades när vi testade att kontrollera vinkeln mellan normalvektorn och vektorn i längsta dimensionens riktning som fås av visionsystemet genom att beräkna den med hjälp av skalärproduktens definition. Slutligen ger inte visionsystemet någon data om vilken plockposition i förhållande till objektets dimensioner som väljs. Under somliga testpackningar hamnade paketen en bit ifrån sin planerade placering vilket efter granskning kunde härledas till att plockpositionen hade valts med samma förskjutning ifrån mittpunkten på paketet. I de flesta fall valdes däremot plockpositionen i mitten på paketets toppyta, vilket gör att objektet kan placeras på rätt plats. Om istället en annan plockposition väljs ges ingen info om detta

och objektet placeras på fel plats.

Vi märkte att visionssystemets förmåga att identifiera olika typer av paket kan påverka valet av plockposition som tidigare berört. Två exempel på problematiska pakettyper är paket med blank yta vars reflektioner stör bildigenkänningen och paket där kontrasten mellan dess färg och plockplatsens omgivning är för låg. Det förstnämnda problemet löses enklast genom att hålla plockplatsen i skugga.

Ett robust placeringssystem ska rimligtvis kunna placera objekten korrekt oberoende av vilken orientering objekten har när de plockas upp. Att möjliggöra detta praktiskt ställer högre krav på visionsystemet så som robotens plockverktyg, vilket är varför detta exjobb inte innefattar en praktisk lösning på detta problem utan enbart en kvalitativ sådan.

Roboten kan enbart rotera föremål runt z-axeln, vilket innebär att den sida på paketet som i placeringsmönstret ska peka nedåt måste peka nedåt vid plocktillfället. Placeringssystemet kan således förbättras genom att införa en ny frihetsgrad vid TCP och rusta pick-and-place-roboten med ett mer avancerat plockverktyg.

4.2 Minizincmodellen

Resultaten visar att MZM klarar av att lösa de vanligaste konfigurationerna på kort tid, oftast under ca 150 ms. Vissa fall, som när vi har förhållandevis många paket tar lite längre tid eller går inte att lösa optimalt under 3 sekunder. Vi kan även konstatera att konfigurationer som innehåller multipler av samma objekt tenderar att ta längre tid. Multiplar resulterar i fler lösnignar med samma kostnad, dvs fler lösnignar för solvern att söka igenom. Försök till att bryta dessa symmetrier gjordes under utvecklingen av minizincmodellen, utan att söktiderna förbättrades. Detta kan undersökas mer i framtida arbeten.

Ett exempel är att lösaren inte klarade av att hitta en unik lösning i tid för några av konfigurationerna med många artiklar kan bero på dels antalet paket men även att det fanns dubletter och tripletter.

Figur 3.3 visar en aspekt av hur lösaren fungerar. Arrangemanget känns inte intuitivt eftersom det är så avlångt. Notera dock att den är volymoptimal, och att det finns extremt lite outnyttjat utrymme ifall det omsluts med en perfekt behållare. I praktiken är det dock inte säkert att en sådan placering är optimal. Kanske är den för instabil. Klarar en så avlång konfiguration ett potentiellt fall under frakt? Hur man ska väga stabilitet och andra praktiska aspekter av en stapling mot volymoptimering är inte självklart och behöver göras först då man har en god förståelse för resterande processer som packningen ska gå igenom. Om man

kan definera det skulle man kunna införa constraints som förhindrar ävlånghet", alternativt vikta optimeringsfunktionen så att avlångheten tas i beaktning.

4.2.1 Marginaler

Constraintlösaren ger en placeringslösning baserat på de uppmätta måtten på paketen. Vi implementerade ett sätt att lägga på marginaler i xy-planet på paketens dimensioner och lät constraintlösaren optimera placeringen givet de förstorade dimensionerna. Vid evaluering av marginalhänsynstagandet vid testning kom vi fram till att det inte var där de stora felen skedde. Placeringen skedde så pass exakt att paketen ytterst sällan stötte i varandra och i de få fall de gjorde det var det så pass lite att de la sig rätt ändå. Hur som helst kan inräknandet av marginaler komma väl till användning utanför testsammanhang, tex som produkt till kund där artikeldatans noggrannhet kanske inte alltid kan verifieras. I figur 3.2 kan vi se ett exempel på en mindre exakt placering som saknar tydlig förklaring. Det skulle kunna bero på allt ifrån att förpackningarna inte är helt symmetriska till att HKM inte släpper ner paketet på ett jämt sätt. I figurens nedre högra hörn är kartongernas kanter räta mot varandra, men om man följer långsidorna uppåt så ökar vinkeln mellan dem. Detta syns dock knappt för ögat vid första anblick när man inte tittar på in inzoomad bild.

4.3 CP och andra metoder

Det finns flera fördelar med CP och att det möjliggör volymoptimala lösningar för placeringsproblemet (givet begränsningarna). Först det, som sagt, ekonomiska och miljömässiga fördelar. Men förutom det finns även en fördel för Cognibotics i att de kan garantera sina kunder denna optimalitet. Det vill säga, att även om en annan lösning, expempelvis ML-baserad, hade presterat i princip lika bra i praktiken, så kan man med den inte garantera optimalitet.

Eftersom MZM i detta arbete kan lösa P:A på en kort tid i sammanhanget (runt 100ms) finns det alltså inte mycket som talar för att en icke-optimal lösningsmetod som ML eller inexakta algoritmer är att föredra i det här fallet.

4.4 Framtida arbeten

Om vi börjar med det praktiska finns det arbete kvar för Cognibotics innan lösaren och systemet kan lanseras för kund. Främst handlar det om att integrera med deras nya visionsystem, säkertställa att en ny version av minizinc kan ”time out” till en suboptimal lösning om en

optimal lösning inte hittas i tid, samt att skriva mer robusta kommunikationsprotokoll och hjälpscipt.

Det finns också mer generella områden det praktiska kan utvecklas på. Exempelvis har vi inte vetskap om något system som kan verifiera en packning/placering. Till exempel om kanterna inte ligger räta mot varandra, som i figur 3.2.

Vidare kan lösaren utvecklas. Om man ska hålla sig till constraint programming kan man säkert utveckla den så att den klarar fler paket på kortare tid än vad vår lösare gör i dagsläget. Det finns också fler praktiska constraints att lägga in för att förbättra lösaren.

Utöver det har Cognibotics (och för andra med linkande placeringsproblem) ett framtida arbete framför sig när det kommer till andra problem. Olika problem kräver olika lösningar, och om Cognibotics vill ligga i framkant så krävs en stor förståelse för placeringsproblem och dess möjliga lösningar i sin helhet.

Kapitel 5

Slutsatser

Packningssystemet och dess lösare är fungerar även om vissa problem återstår innan systemet kan användas i produktion. Lösaren visade sig ha bra prestanda för de vanligaste konfigurationerna, men vissa fall tog längre tid att lösa, särskilt de med flera identiska objekt. Hur som helst har lösaren testats på många av casets vanligaste scenarion och presterat bra vilket tyder på att det är en lämplig lösare för det specifika fallet. Framtida arbeten föreslås för att göra systemet mer robust och förbättra eller vidareutveckla lösaren utifrån alla aspekter.

5.1 Svar på forskningsfrågor

I rapportens inledning ställdes tre forskningsfrågor. Här sammanfattas svaren till dessa i korthet.

Vad finns det för existerande lösningar till 3D-placeringsproblem?

Först kan man dela upp lösningarna till placeringsproblem i exakta/optimala lösningsmetoder och inexakta/inoptimala. CP möjligör en exakt lösning eftersom den metoden tillåter oss att söka igenom alla möjliga lösningar, givet begränsningarna, och sedan välja den bästa. Inexakta lösningsmetoder är exempelvis ML-baserade, eller algoritmer som inte kan garantera optimala resultat.

Är constraintprogrammering en lämplig metod för att lösa 3D-placeringsproblem?

Constraintprogrammering är en lämplig metod för att lösa åtminstone problemet som behandlas i det här arbetet. För mer komplexa problem, exempelvis de som involverar fler objekt som ska placeras, kan andra metoder vara mer lämpliga. Det finns många fall där former är mer komplexa, och svårare att packa, än rätblock. Alltifrån cylindrar till former som är väldigt svåra att ens stapla, som vitlökar. Det är möjligt att CP, kan användas för att lösa även dessa uppgifter, exempelvis genom att approximera dem som rätblock och markera som ostaplingsbar i Minizinc. Då riskerar man dock att förlora garantin på optimalitet.

Vilka svårigheter finns i att omsätta en teoretisk placering till ett fungerande system för HKM?

Den främsta svårigheten för det här arbetet var kvaliteten på visionsystemet. Ett problem med dess djupseende gjorde att vinklar inte kunde beräknas korrekt vilket ledde till att vinkelkorrigering inte kunde testas i praktiken. Bildanalys för att hämta plockpositioner var dessutom väldigt långsam och gjorde att systemet i sin helhet inte kunde köras i speciellt hög hastighet. Vidare påverkas systemet av begränsningar i frihetsgraderna som nämns i avsnitt 2.1.1. Enligt dessa tillåts objekt inte rotera längs z-axeln och i detta arbete har vi därför förutsatt att objekten har korrekt z-orientering vid plocktillfället. Huruvida Cognibotics kund har möjlighet att säkerställa att paketen har korrekt z-koordinat vid plocktillfälle för att uppnå en sömlöshet i systemet är ej fastställt och en potentiell lösning hade därför kunnat vara att införa begränsningar på lösningarna som gör att man låser z-koordinaten i lösaren till orienteringen paketet har när det identifieras av visionsystemet men är inget som utforskats i detta arbete.

Litteratur

[1] Peter van Beek, Francesca Rossi och Toby Walsh. "Handbook of Constraint Programming". I: (2006). URL: https://books.google.se/books?id=Kjap9ZWcKOoC&q=handbook+of+constraint+programming&pg=PP1&redir_esc=y#v=onepage&q=constraints&f=false.

[2] Andreas Bortfeldt och Gerhard Wäscher. "Constraints in container loading–A state-of-the-art review". I: *European Journal of Operational Research* 220.2 (2012). ISSN: 0377-2217. URL: https://www.sciencedirect.com/science/article/pii/S037722171200937X.

[3] Cognibotics. *Cognibotics: About Us*. https://cognibotics.com/about-us/. Accessed on October 11, 2023.

[4] Cognibotics AB. *HKM1800*. 2023. URL: https://cognibotics.com/hkm (hämtad 2023-11-23).

[5] Jeff Erickson. *Algorithms*. 1st. 2019, s. 381.

[6] European Commission. *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on packaging and packaging waste, amending Regulation (EU) 2019/1020 and Directive (EU) 2019/904, and repealing Directive 94/62/EC*. Proposal COM(2022) 677 final. European Commission, 2022.

[7] Daniel Hellström. "On interactions between Packaging and Logistics - Exploring implications of technological developments". I: (2007). URL: https://portal.research.lu.se/en/publications/on-interactions-between-packaging-and-logistics-exploring-implica.

[8] *JaCoP - Java Constraint Programming Solver*. https://www.minizinc.org/challenge2022/description_jacop.txt. Accessed on January 11, 2024. MiniZinc, 2024.

[9] KEBA AG. *KEStudio Engineering*. 2023. URL: https://www.keba.com/en/industrial-automation/products/software/kestudio-engineering (hämtad 2023-06-28).

[10] Krzysztof Kuchcinski. "Modeling and Optimization of embedded system with constraint programming: principles and practice". I: (2020).

[11] Damien Lopez. *Who wants to figure out how to pack anyway?* Medium. Blog post. Maj 2022. URL: https://www.anylogic.com/blog/solving-the-bin-packing-problem-in-warehousing-and-logistics-strategy-comparison/ (hämtad 2023-06-28).

[12] Silvano Martello, David Pisinger och Daniele Vigo. *The Three-Dimensional Bin Packing Problem*. Tekn. rapport DEIS - OR - 97 - 6. Operations Research. Viale Risorgimento, 2, 40136 Bologna, Italy: Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi di Bologna, maj 1997.

[13] Naturvårdsverket. "Energins påverkan på miljön". I: (2023). URL: https://www.naturvardsverket.se/amnesomraden/klimatomstallningen/omraden/klimatet-och-energin/energins-paverkan-pa-miljon/#:~:text=Milj%C3%B6p%C3%A5verkan%20fr%C3%A5n%20fossila%20br%C3%A4nslen,och%20mark%20och%20orsakar%20h%C3%A4lsoproblem..

[14] Nicholas Nethercote m. fl. "MiniZinc: Towards A Standard CP Modelling Language". I: (). URL: https://www.minizinc.org/pub/nethercote_et_al_minizinc.pdf.

[15] Pontus Haglund. *Deklarativ programmering, TPD007 Konstruktion av datorspråk, föreläsning 7*. Tekn. rapport. Instutitionen för datavetenskap vid Linköpings Universitet, 2023.

[16] Peter J. Stuckey, Kim Marriott och Guido Tack. *The MiniZinc Handbook*. Accessed on January 11, 2024. Chapter 3.4 - Solving Technologies and Solver Backends. 2016–2020. URL: https://www.minizinc.org/doc-2.5.5/en/solvers.html?highlight=gecode.

[17] K. Wang m. fl. "K-ST: A Formal Executable Semantics of the Structured Text Language for PLCs". I: (2023). URL: https://search-ebscohost-com.ludwig.lub.lu.se/login.aspx?direct=true&AuthType=ip,uid&db=edseee&AN=edseee.10251676&site=eds-live&scope=site.

[18] Hang Zhao m. fl. "Online 3D Bin Packing with Constrained Deep Reinforcement Learning". I: *AAAI Conference on Artificial Intelligence* (2021), s. 1–3.

Institutionen för Datavetenskap | Lunds Tekniska Högskola | Redovisas 31 Augusti 2023
Examensarbete: Utveckling och Implementering av artikelplaceringssystem för industriell robot med constraintprogrammering
Student(er): Astrid Dymling, Martin Nybleus
Handledare: Per Andersson (LTH), Olle Hedbrant (Cognibotics)
Examinator: Flavius Gruian

Automatiserad packning för E-handel

Populärvetenskaplig Sammanfattning av ***Astrid Dymling, Martin Nybleus***

I industriella sammanhang är packning en återkommande utmaning. Platseffektiv packning och paketering möjliggör både ekonomiska och miljömässiga besparingar. I en perfekt värld hade varje paket som skickas med posten, varje lastutrymme i lastbilar och varje container i lastfartyg varit optimalt packade, det vill säga helt utan luft! Robotföretaget Cognibotics vill alltså självfallet kunna garantera sina kunder att deras plock- och placeringsrobot HKM kan prestera optimalt inför alla typer av packningsutmaningar. Men hur packar man egentligen en order optimalt? Vilken matematik ligger bakom? Och klarar HKM:en av uppgiften?

Miljöpåverkan och EU-regelverk

Den miljömässiga vikten av att minimera onödigt tomrum under transport diskuteras till och med i EU. Begreppet ”tomrum” inkluderar inte bara fysiskt tomt utrymme, utan även onödig användning av fyllnadsmaterial. För att ta itu med detta har Europeiska kommissionen förberett ny lagstiftning som begränsar andelen tomrum i olika typer av frakter, inklusive e-handelspaket, till 40 procent av paketets totala volym. Företag som är beroende av e-handel kommer behöva anpassa sina förpackningsmetoder för att följa de nya reglerna.

Olika typer av packningsproblem

Det finns flera olika typer av packningsproblem och väldigt många unika fall. Det kan handla om att en orders ingående artiklar är kända på förhand eller att de kommer på löpande band. Kanske är artiklarna perfekta klossar, som är enklare att arbeta med matematiskt eller så har de mer komplexa former.

Matematiken bakom

Matematiskt sett kallas den här typen av problem för 3D-bin packing. Med bin menas ett rätblock. Om ett föremål i verkligheten inte är ett rätblock approximerar vi det alltså som det. Inom 3D bin packing är det avgörande hur många paket som ska packas. För fler än ca 20 paket kan man exempelvis inte vara säker på att man hittat den bästa lösningen, om man inte har väldigt lång tid på sig! Den typen av lösningar kallar man för “inexakta”. För Cognibotics och HKM, som i dagsläget ska packa upp till 10 stycken på förhand kända paket, spelar det dock ingen roll. Då är det oftast möjligt att hitta exakta lösningar. En metod man kan använda för att hitta exakta lösningar är “constraint programming”, vilket är den metoden vi använde. Med constraint programming kan man definera ett problem och sedan låta programmet leta igenom alla möjliga lösningar och spara den bästa. Detta tar lång tid, men vi lyckades konstruera vårt program så att det går tillräckligt snabbt för upp till ca 10 paket.

Kan roboten packa nu då?

Ja det kan den! Och med lite finjusteringar kommer den snart att vara redo att packa ditt paket på bästa möjliga sätt nästa gång du beställer hem något.